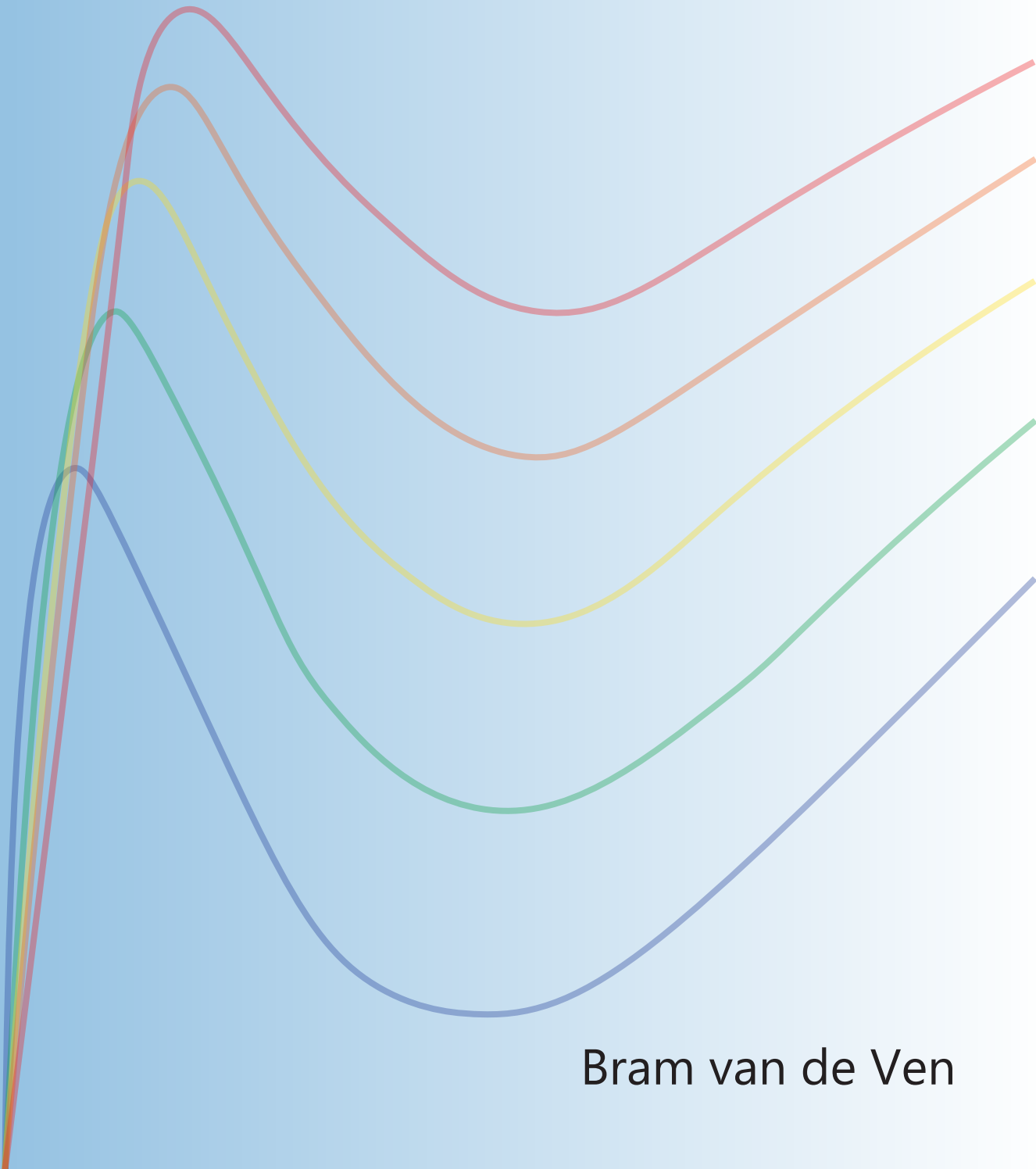
Classification using Dopant Network Processing Units



Bram van de Ven

Classification using Dopant Network Processing Units

Bram van de Ven

The research described in this thesis was carried out in the NanoElectronics group at the MESA+ Institute for Nanotechnology at the University of Twente, Enschede, The Netherlands, and was financially supported by the Netherlands Organization for Scientific Research (NWO).

UNIVERSITY OF TWENTE. | MESA+ INSTITUTE

Graduation Committee:

Chairman & Secretary:
Prof. Dr. J.N. Kok

Supervisor:
Prof. Dr. ir. W.G. van der Wiel — University of Twente

Co-supervisor:
Prof. Dr. P.A. Bobbert — University of Twente

Other members:
Prof. Dr. ir. H.J. Broersma — University of Twente
Prof. Dr. ir. J.W.M. Hilgenkamp — University of Twente
Prof. Dr. T. Banerjee — University of Groningen
Prof. Dr. S. Stepney — University of York

Title: Classification using Dopant Network Processing Units

Author: Bram van de Ven

Cover Design: Bram van de Ven

Printed by: Gildeprint

ISBN: 978-90-365-5373-5

DOI: 10.3990/1.9789036553735

Classification using Dopant Network Processing Units

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
Prof. Dr. ir. A. Veldkamp,
on account of the decision of the graduation committee,
to be publicly defended
on Wednesday the 15^{th} of June 2022 at 14:45

by

Bram van de Ven
born on the 23^{rd} of September 1994
in Veldhoven, The Netherlands

This dissertation has been approved by:

Supervisor

Prof. Dr. ir. W.G. van der Wiel

Co-supervisor:

Prof. Dr. P.A. Bobbert

Contents

1

Introduction

Artificial intelligence (AI) is becoming increasingly important in everyday life. It has led to applications such as face recognition[1] and virtual personal assistants in the form of Siri, Alexa and Cortana[2]. AI is even capable of outperforming humans at complex games such as Go[3] and Dota2[4]. In general, AI mimics the problem solving and decision making capabilities of the human mind. A subdiscipline of AI is machine learning (ML). ML focuses on developing systems that are capable of deriving a desired behaviour from data. Among the most common implementations of ML, artificial neural networks (ANNs) are commonly used. These brain-inspired networks can be trained towards a specific functionality. The relation between AI, ML and ANNs is illustrated in Figure 1.1a. In general, ANNs are implemented to perform tasks that are difficult to solve by hard coding[5], such as face recognition[1].

One of the first artificial neuron (AN) models was already proposed in the 1940s when McCulloch and Pitts defined the MP (McCulloch-Pitts) neuron. This neuron takes binary inputs, of which the sum is compared with a threshold to give a 1 or 0 output[6]. In the 1950s, Rosenblatt expanded this concept by allowing real-valued inputs that are subsequently weighted (linearly multiplied by a so-called weight value). This AN was the building block for the first ANNs, also known as the perceptron[7]. In the 1960s these building blocks were implemented in the first ANNs[8]. At this point, ANN training was limiting their size. Techniques now used in training ANNs, such as backpropagation[9], were already developed at that time. However, until the late 1980s, these techniques were only used for small networks[8]. Nowadays computers are significantly more powerful than 40 years ago. This has led to the realisation of large ANNs that, together with the gradient descent[10] learning approach, are leading ML applications[11].

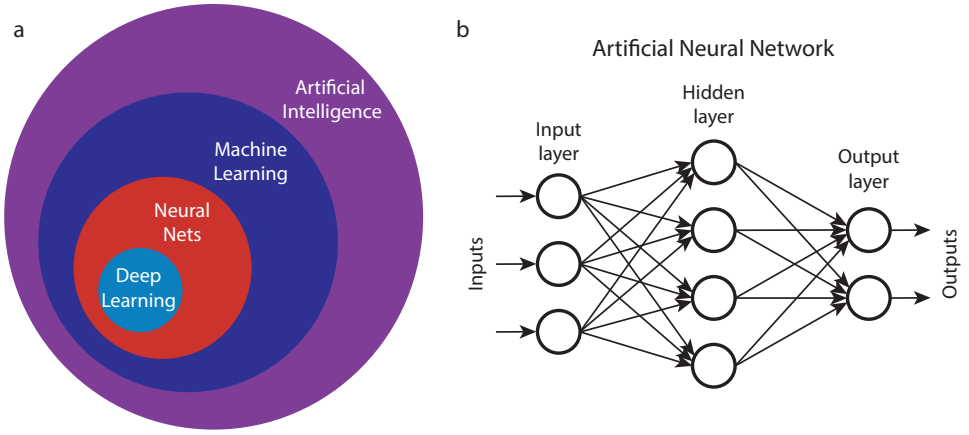


Figure 1.1: a, Overview of the relation between artificial intelligence (AI), machine learning (ML), neural networks (NNs) and deep learning (DL). b, Schematic representation of a feed-forward artificial neural network (ANN), where the black arrows indicate the weights applied to the outputs of the nodes in the previous layer.

Figure 1.1b shows an illustration of a small ANN with only a single hidden layer (a layer of nodes between the input and output layer). The black lines indicate the weights of a multiply and accumulate (MAC) operation, including a bias. The nodes in the hidden layer represent a nonlinear activation function. Many of the real-world implementations of ANNs are realized using deep learning (DL). As illustrated in Figure 1.1a, DL is a subdiscipline of ANNs. An ANN becomes a so-called deep neural network (DNN) when it has multiple hidden layers[12]. When such a network is large enough, it could theoretically be used as a universal function approximator, meaning that the multiplication weights and biases, in the MAC layers, can be tuned such that the network reproduces any possible input-output relation.[13]. When learning, for example, to recognise your face, the weights and biases of the MAC layers are optimised such that the network performs the function that links the image of your face to you (unlocking your phone). However, as one can imagine, creating very big DNNs that can perform very complex tasks requires storing many weights and biases and performing numerous MAC operations. Both of these currently are limiting factors to the size of DNNs.

Conventional digital computers, used to implement DNNs, are limited by the Von Neumann architecture[14]. In this architecture memory and processing are separated. Due to the large number of weights and biases stored when using a DNN, the separation between the processor and memory units requires substantial data transfer. However, transferring data is energy-intensive and relatively slow, limiting the size of efficient DNNs[15]. At the same time, the increasing computing requirements of

DNNs also come with the need for more processing power. Regarding the processing power of conventional computers, Moore's law states that the number of transistors in integrated circuits (ICs) doubles every two years[16]. This law has been the driving force behind the IC. However, we are getting closer to the physical limits where a single transistor reaches the size of a few atoms. Both the data transfer and processing power limitations of ICs have given rise to the field of neuromorphic engineering[17].

1.1 Neuromorphic Engineering

The term neuromorphic engineering describes the development of hardware that is inspired by the structure of the brain. Based on the neurons in the brain, physical artificial neurons (ANs) are being developed. When connected in a network, physical ANs are capable of basic AI functionality. Such neuromorphic hardware needs to be trainable and aims to exploit parallelism and in-memory computing[18]. The data of neuromorphic hardware are mostly presented using one of two methods. The first method, concerns neurons for which the datasets used are introduced as numerical values for analogue computation[19]. In the second method, the information is encoded in the timing and amplitude of spikes[20]. These spiking ANs are inspired by the spike-based communication in the brain. Due to their event-based nature, spikes are very energy efficient since the signal is not always on. Because of this efficiency, many neuromorphic hardware implementations utilise spikes[8].

There are quite a few neuromorphic engineering approaches that use standard technologies to implement physical ANs. These ANs can be implemented in the digital, analogue or hybrid domain. In the digital domain, designing neurons is easier since they are more robust, while analogue neurons could perform more efficient computing[21]. The robustness and simplicity of the digital approach lends itself well to big companies such as IBM, which developed TrueNorth, one of the first neuromorphic chips in 2014[22]. This thumb-sized chip implements a network of a million spiking neurons in an integrated circuit. Later, Intel followed the digital path with Loihi (2018)[23] and Loihi2 (2021)[24]. Loihi2 also has a million neurons but claims to be faster in both processing and communication as compared to previous digital neuromorphic chips[25]. The promise of efficient computing has led researchers to focus more on the analogue domain. This research aims to implement a network of memory elements capable of parallel information processing. One approach utilises flash memory for the memory part of an artificial neuron[26]. The advantage of using flash technology is the existing knowledge from its implementation in data storage. Especially in the analogue domain, neuromorphic engineers are looking for novel materials to implement physical ANs.

A class of devices based on novel materials are memristors. The standard memristor implements the memory component of a physical AN in programmable non-volatile resistance states. In terms of the ANN in Figure 1.1b, these resistance states achieve an analogue implementation of the MAC operations (black lines)[27]. Another approach uses, instead of neurons, physical disordered dynamical systems. Due to the complex time-dynamics nature of these systems, they are difficult to train. Therefore, these dynamic systems are often combined with reservoir computing (RC)[28]. RC uses many states (outputs) from the dynamic system and linearly maps them to a desired output. In this case, only the linear readout layer needs to be trained, which reducing the complexity of the training process. Hence, using RC the computational capabilities of any complex dynamic system can be exploited[29].

Another method to exploit novel materials to achieve AI functionality uses intelligent matter[30]. Intelligent matter needs to respond to external stimuli and be trainable to use these stimuli to exhibit the desired behaviour. One approach that uses intelligent matter is evolution-in-materio. Evolution-in-materio uses a so-called genetic algorithm (GA) to train complex material systems[31]. This approach to utilising complex material systems was implemented using gold nanoparticles in 2015, the nanoparticle network was trained to perform Boolean logic as a benchmark[32]. In this thesis, we build on this work by utilizing dopant network processing units (DNPUs), networks of dopants in a semiconductor that, when operated at the correct temperature and dopant concentration, can be used for evolution-in-materio[33]. The DNPUs in this thesis are made from boron/arsenic doped silicon. Furthermore, we show that DNPUs can be interconnected to perform more complex tasks[34]. By utilising the RC framework with DNPUs, we show that it is possible to extract more complex nonlinear behaviour. Moreover, we use this behaviour to implement the required nonlinearity of physical ANs in hardware, where a digital computer performs all linear operations. Finally, we show that the global tuneability combined with a flexible geometry of DNPUs allows us to reduce the number of stored parameters, and therefore memory requirements, needed to perform nonlinear tasks using ANs. Using this material platform as an example, we emphasize the importance of researching novel material systems to develop intelligent matter capable of both learning complex behaviour and incorporating the memory required to store the learned parameters.

1.2 Outline of the Thesis

This thesis is structured as follows.

Chapter 2 gives an overview of the, for this thesis, relevant research performed in the field of novel materials for neuromorphic engineering. Here we discuss memristors that implement the MAC operations of physical ANs, reservoirs that implement disordered dynamical networks and evolution-in-materio. Finally, we elaborate on how the physical properties of DNPUs are useful to implement evolution-in-materio.

In Chapter 3 the fabrication processes of DNPUs are described. We separately discuss the micron-scale (wafer) processes centered around photolithography and the nano-scale (chip) processes centered around electron beam lithography. The micron-scale processes allow for quick fabrication of repetitive structures while the nano-scale processes allow for a high degree of flexibility. After the fabrication sections, we describe the measurement set-ups used to characterise DNPUs.

The first electrical characterisation of DNPUs is presented in Chapter 4. We start by analysing the temperature dependence of the conduction and find that, when cooling the DNPUs, they exhibit non-linear current-voltage, I-V, characteristics. By further analysis we show that between 70-160 K (where the non-linear conduction takes place) variable range hopping (VRH) is the dominant conduction mechanism. We demonstrate evolution-in-materio using the reconfigurable Boolean logic benchmark. Furthermore, we expand on this by training the DNPUs to extract the features of 2x2 pixel maps. Finally, by using the input-output relations measured on the DNPUs for these pixel maps, we show, by simulation, that the MNIST handwritten digit recognition benchmark can be solved with an accuracy of 96 % by using a network of multiple DNPUs in parallel.

Since obtaining functionality is limited by the training of the DNPU, we present a new method for training nano-electronic devices in Chapter 5. We show that it is possible to train a DNN that will have the same input-output behaviour as the DNPU. This DNN model is called a surrogate model (SM) of the DNPU. Using such a SM, it is possible to use standard deep learning techniques, such as gradient descent, to find the desired tuning parameters (control voltages) of the DNPU. By applying these voltages on the physical device we validate the behaviour of the SM on a physical DNPU. Using this training method, we show that DNPUs are capable of mapping any 2x2 pixel map to a unique discrete current level.

In Chapter 6, we analyse the computational capabilities of a single DNPU using the Vapnik-Chervonenkis dimension (VC dimension), which quantifies the number of binary tasks that the DNPU can solve. After this, using a single SM of the DNPU, a network of SMs is made and trained. The behaviour of this network can be verified by measuring the physical DNPU, from which the SM is derived, multiple times using

the sets of control voltages found from training the SM (one set per SM used in the network) to emulate the behaviour of a network of DNPUs. We compare a network of DNPUs connected in a 2-2-1 structure, where the numbers "2" and "1" indicate how many DNPUs are placed in parallel. This comparison shows that the 2-2-1 network outperforms the single DNPU, giving an indication of the potential to scale towards networks of DNPUs to perform more complex tasks.

In Chapter 7, we discuss the extraction of more computational power from a single DNPU by operating it in a different mode (configuration). We move away from the 1-output mode that is mostly used and show that, for a DNPU with 12 electrodes, there is an operation mode in which more nonlinearity can be extracted(2 inputs, 2 controls, and 8 outputs electrodes). Using the DNPU in this mode, we solve the Hillenbrand vowel recognition benchmark with an accuracy of 89.9 %, by emulating the behaviour of a network of multiple DNPUs in parallel. The network of DNPUs requires less stored parameters than a minimal ANN with the same number of inputs and outputs, showing the advantage of the global tuneability of DNPUs.

Finally, in Chapter 8, we analyse the potential for DNPUs for neuromorphic computing. It seems possible to create a network of interconnected DNPUs. By combining DNPUs with other linear technologies, the complex behaviour of the DNPUs can perform the nonlinear operations and take over some of the linear operations. This complementary approach is the most promising path when using DNPUs for neuromorphic computing.

References

[1] L. Li, X. Mu, S. Li, and H. Peng. A review of face recognition technology. *IEEE Access*, 8:139110–139120, 2020. doi: 10.1109/ACCESS.2020.3011028.

[2] V. Këpuska and G. Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 99–103, 2018. doi: 10.1109/CCWC.2018.8301638.

[3] R. John. Google ai beats go world champion again to complete historic 4-1 series victory, 2016. URL `https://techcrunch.com/2016/03/15/google-ai-beats-go-world-champion/`. Last accessed 11 February 2022.

[4] OpenAI Blog. Dota 2, 2017. URL `https://openai.com/blog/dota-2/`. Last accessed 11 February 2022.

[5] A. Athem. *Machine Learning.* The MIT Press, 2021.

[6] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943. doi: 10.1007/BF02478259.

[7] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. doi: 10.1037/h0042519.

[8] Y. Chen, H. Li, C. Wu, C. Song, S. Li, C Min, H. Cheng, W. Wen, and X. Liu. Neuromorphic computing's yesterday, today, and tomorrow – an evolutional view. *Integration*, 61:49–61, 2018. doi: 10.1016/j.vlsi.2017.11.001.

[9] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *International 1989 Joint Conference on Neural Networks*, pages 593–605 vol.1, 1989. doi: 10.1109/IJCNN.1989.118638.

[10] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv*, 2017. doi: 10.48550/arXiv.1609.04747.

[11] M. Davidson. Review on gradient descent algorithms in machine learning. *Journal for innovative development in pharmaceutical and Technical Science*, 2:88–93, 2019.

[12] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. doi: 10.1038/nature14539.

[13] S. Sonoda and N. Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43 (2):233–268, 2017. doi: 10.1016/j.acha.2015.12.005.

[14] P. P. Jagtap. Embedded systems architecture-review. *IRJET*, 5(11):227–229, 2018.

[15] P. M Kogge. Function-based computing and parallelism: A review. *Parallel Computing*, 2(3):243–253, 1985. doi: 10.1016/0167-8191(85)90006-7.

[16] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), 1965.

[17] J. Shalf. The future of computing beyond moore's law. *Phil. Trans. R. Soc. A.*, 378(2166), 2020. doi: doi.org/10.1098/rsta.2019.0061.

[18] S. Soman, jayadeva, and M. Suri. Recent trends in neuromorphic engineering. *Big Data Analytics*, 1(1):15, Dec 2016. doi: 10.1186/s41044-016-0013-1.

[19] M. Hu, H. Li, Q. Wu, and G. S. Rose. Hardware realization of bsb recall function using memristor crossbar arrays. In *DAC Design Automation Conference 2012*, pages 498–503, 2012. doi: 10.1145/2228360.2228448.

[20] S. M. Schuetze. The discovery of the action potential. *Trends in Neurosciences*, 6:164–168, 1983. doi: 10.1016/0166-2236(83)90078-4.

[21] A. Joubert, B. Belhadj, O. Temam, and R. Héliot. Hardware spiking neurons design: Analog or digital? In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5, 2012. doi: 10.1109/IJCNN.2012.6252600.

[22] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, S. Cassidy A, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. doi: 10.1126/science.1254642.

[23] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

[24] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. Bam Shrestha, F. T. Sommer, and M. Davies. Efficient Neuromorphic Signal Processing with Loihi 2. *arXiv*, 2021. doi: 10.48550/arXiv.2111.03746.

[25] Intel. Loihi 2: A new generation of neuromorphic computing, 2022. URL `https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html`. Last accessed 11 February 2022.

[26] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov. High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4782–4790, 2018. doi: 10.1109/TNNLS.2017.2778940.

[27] Q. Xia and J. J. Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18(4):309–323, Apr 2019. doi: 10.1038/s41563-019-0291-x.

[28] H. Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. *GermanNational Research Center for Information Technology*, 2002.

[29] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019. doi: 10.1016/j.neunet.2019.03.005.

[30] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice. The rise of intelligent matter. *Nature*, 594(7863):345–355, Jun 2021. doi: 10.1038/s41586-021-03453-y.

[31] J.F. Miller and K. Downing. Evolution in materio: looking beyond the silicon box. In *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, pages 167–176, 2002. doi: 10.1109/EH.2002.1029882.

[32] S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature Nanotechnology*, 10(12):1048–1052, Dec 2015. doi: 10.1038/nnano.2015.207.

[33] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.C. Ruiz Euler, H. Broersma, P.r A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel. Classification with a disordered dopant-atom network in silicon. *Nature*, 577(7790):341–345, Jan 2020. doi: 10.1038/s41586-019-1901-0.

[34] H.C. Ruiz-Euler, U. Alegre-Ibarra, B. van de Ven, H. Broersma, P. A. Bobbert, and W. G. van der Wiel. Dopant network processing units: towards efficient neural network emulators with high-capacity nanoelectronic nodes. *Neuromorphic Computing and Engineering*, 1(2):024002, sep 2021. doi: 10.1088/2634-4386/ac1a7f.

2

Novel Materials for Neuromorphic Engineering

In the field of neuromorphic engineering, researchers are developing physical systems to perform brain-inspired computations[1]. Brain-inspired computation does not mean that we aim to reproduce all detailed aspects of the brain. Instead, a more common approach is to implement aspects of the basic building blocks of the brain, such as neurons, tuneable synapses and spikes[2]. Such artificial neurons (ANs) need to be able to work together to form complex networks capable of neuromorphic computing[3]. A neuromorphic computer is a system that mimics the architecture of neuro-biological systems to perform computation. When designing hardware to implement a neuromorphic computer, there are three requirements. First, the physical ANs in the system need to be responsive to external stimuli such that the external information is processed. Second, the ANs need to achieve the desired response from external stimuli. Hence, they need to be trainable. Third, the ANs need to store the learned state in a non-volatile memory[4].

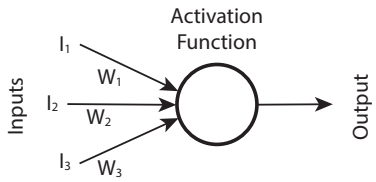


Figure 2.1: Schematic representation of a feed-forward artificial neuron as implemented in digital computers.

To better understand the different approaches to implementing partial AN functionality, we first look at how ANs are implemented in the software approach to creating artificial neural networks. An illustration of an AN, with inputs I_n, is shown in Figure 2.1. There is no limitation concerning the number of inputs. The training and memory are implemented in the weights (W_n)[5]. Each input has a weight by which it is multiplied. These $I_n \cdot W_n$ multiplications are accumulated to perform a multiply-and-accumulate (MAC) operation[6]. The output of the MAC operation is passed through a nonlinear activation function to allow a network of such neurons to learn tasks beyond matrix multiplication. The activation function in the AN can vary from a step function [7] to a sigmoid[8] and even a Rectified Linear Unit (ReLu)[9] can be used[10].

Novel materials present an interesting opportunity to implement different aspects of artificial neurons. Because Artificial Neural Networks (ANNs) are limited by the Von Neumann bottleneck[11, 12], researchers have tried implement the weights of the AN in Figure 2.1 using non-volatile materials such that networks can be created that perform the MAC operations in hardware for in-memory computing[13]. It is also possible to create disordered material systems that exhibit nonlinear behaviour. Often these materials have a high degree of nonlinearity. Combining this with the requirement that the system needs to be sensitive to external stimuli, these material systems can be trained for brain-inspired tasks. In this case, such networks can be seen as a combination of physical "ANs" that implement a neuromorphic network[14, 15, 16]. This chapter will illustrate how novel material systems are used for neuromorphic engineering.

2.1 Novel Materials for In-memory MAC Operations

To implement in-memory MAC operations, the material needs to exhibit multiple non-volatile states that linearly respond to external stimuli. In other words, the material needs to be capable of both storing and processing the weights of an AN[13]. Networks performing MAC operations are achieved in both the electronic[17] and optical domain[18].

2.1.1 MAC operations using Memristors

One method to perform in-memory MAC operations uses memristive devices. In these devices, weights are programmed as different conductance states of the memristive material. These states can interact with external voltages. Following Ohm's law, an applied voltage is multiplied by the conductance of the material, resulting in a current. In this section we will describe conductive-filament[19] and phase-change memory (PCM)[20] based memristors[17].

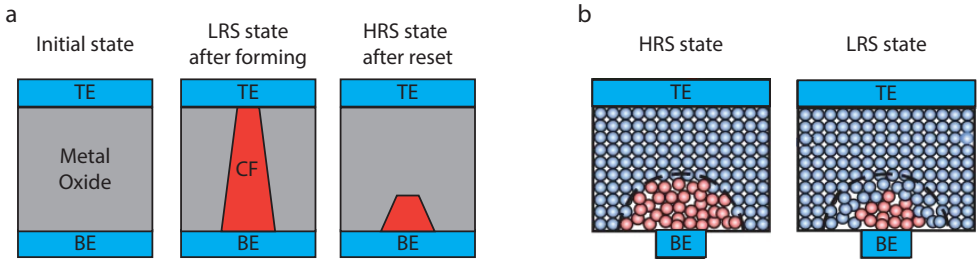


Figure 2.2: Illustration of the working principle of memristors with the top and bottom electrodes (TE, BE) in blue. a, The formation of a conductive filament (CF, red) in a metal oxide memristor with the initial state (left), low-resistance state (LRS, middle) and high-resistance state (HRS) after the first reset (right) [21].b, A PCM memristor showing both the HRS (left) and LRS (right)[20].

Conductive-filament memristors are usually fabricated by making a metal/insulator/metal stack[13]. In many cases the insulator is a metal oxide such as TiO_x[22], TaO_x[23] and HfO_x[24]. In these devices, the oxygen vacancies can move under the influence of an electric field, forming a conductive filament that changes the conductance of the metal oxide material[13, 25]. For these materials, the conductance can be tuned by adjusting the size of the filament. A schematic illustration of this process is shown in Figure 2.2a. In 2020 such memristors were used to create a network of multiple memristors, together capable of solving the MNIST handwritten digit classification task. These conductive-filament memristors used a TiN/TaO_x/HfO_x/TiN stack to program the different weight values[26].

Phase-change memristors achieve the different conductance states due to the different conductivities of the material's crystalline vs amorphous state. By heating up the PCM material using different voltage pulse heights and widths, the ratio of amorphous to crystalline material can be adjusted[27]. Based on this ratio the PCM will have a well-defined non-volatile conductance state. An illustration of a PCM device is shown in Figure 2.2b. Such PCM memristors have been used to create arrays that perform MAC operations[28]. The advantage of PCM memristors is that the behaviour of PCM is well understood, making it easier to engineer the devices to smaller structures[29].

2.1.2 MAC Operations using Optical Networks

Another approach to implement MAC operations using novel materials can be performed in the optical domain. When implementing MAC operations in the optical domain, microscale optical frequency combs (microcombs) are often used. These combs allow different information to be programmed and addressed at different frequencies in the signal. This approach is inherently parallel, allowing for scalability

towards many operations[30]. By adding a waveshaper, the optical power of the signal emanating from the microcomb can be adjusted. This microcomb-waveshaper combination has been implemented in two ways. In the first method, the microcomb and waveshaper define the input signal to which the weights are applied[31]. In the second method, the microcomb and waveshaper are used to define the weights that are used to interact with the input data[32].

The first method again uses PCM to store the weights. By fabricating a waveguide with a small PCM area on top, the evanescent field of the light couples to the PCM. Based on the ratio of crystalline to amorphous material in the PCM, it will absorb a different amount of optical power, making it possible to perform multiplications based on the energy in the optical signal[33]. This effect is illustrated in Figure 2.3a. Using a network of multiple waveguides, each with their own PCM to store different weights, it has been shown that to be possible to implement non-volatile MAC operations in the optical domain [31, 34].

The second method uses the microcomb and waveshaper to define the weights of the MAC operation. By using an electro-optical Mach-Zehnder modulator[35] to create a linear interaction between these weights and electrically programmed input, it is possible to perform MAC operations. This approach was first used to create an optical AN[32] (see Figure 2.3b) and later expanded by creating an optical network capable of large scale MAC operations where the waveshaper stores the required weights[36].

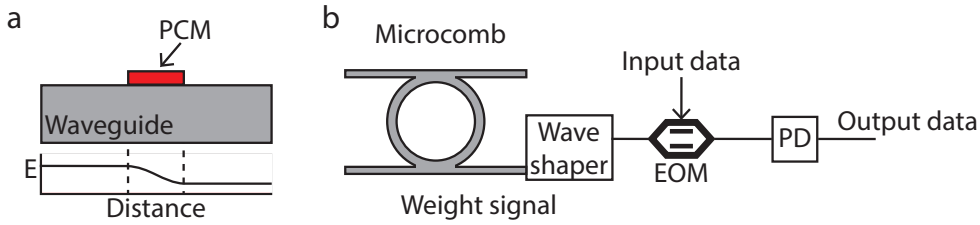


Figure 2.3: Illustration of the two optical multiplication approaches. a, Waveguide with the PCM (top) changing the light transmission through the waveguide[33]. b, Multiplication of the weight signal, defined by the microcomb and waveshaper (left), with the input data using an electro-optical modulator (middle) followed by conversion to an electrical signal using a photodetector (PD, right)[32].

2.2 Disordered Material systems for Neuromorphic Engineering

The complexity present in disordered material systems has inspired researchers to use these materials for computation. This has given rise to two approaches, physical reservoir computing[37, 38] and evolution-in-materio[39]. The physical reservoir computing approach extracts the nonlinear behaviour of the material's system without tuning it. The physical reservoir states are linearly combined to perform neuromorphic tasks. Evolution-in-materio uses an evolution-inspired training technique to tune the material system to perform such tasks. Here the output of the material is directly used for the neuromorphic tasks. In this section, we describe both methods and give some examples of material systems used to implement these concepts.

2.2.1 Physical Reservoir Computing

The physical reservoir computing (RC) approach to utilising disordered materials systems is inspired by the RC approach to training recurrent neural networks (RNNs). RNNs are ANNs implemented in software that have recurrency in the way they are connected. This results in data retention in the network allowing for dynamic tasks such as time-series prediction[40]. At the same time, this recurrency makes RNNs difficult to train. Hence, RC is used to exploit the dynamical behaviour of RNNs while still keeping the training simple[41]. RC treats the RNN as a reservoir without any tuning. The inputs are mapped onto the internal reservoir states. In turn, the reservoir states are mapped onto outputs by a trainable linear layer to perform the desired computation.[42]. This process is illustrated using an RNN in Figure 2.4a.

Physical reservoirs exploit material systems that inherently have similar properties as a RNN. This material system is used to map inputs to a set of output states. Just like in RC, the training is done by finding a linear layer that is capable of mapping the output states of the material to the desired functionality[37, 38, 43]. This approach is illustrated in Figure 2.4b. Such a material-based reservoir needs to

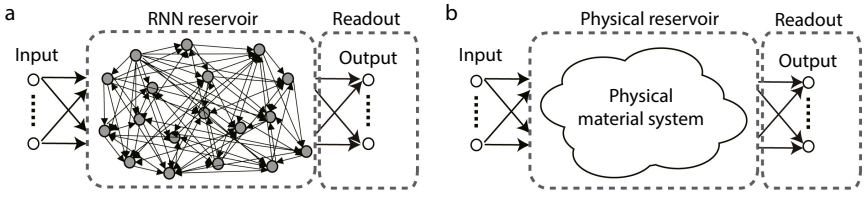


Figure 2.4: a, Schematic representation of a reservoir computing system with a recurrent neural network (RNN) reservoir[37]. b, Schematic representation of a reservoir computing system with a physical material based reservoir[37].

have three properties. The first is memory (memory capacity). Since RC is mainly used for dynamical tasks, it is important that the material has some dependence on its previous states. The second requirement is generalisability (quantified by a generalisation rank). This requirement states that, with the same input information, the reservoir should provide similar information in its reservoir states. This limits the noise and requires a method to reset the memory back to its initial state. Lastly, the reservoir needs to have a certain amount of non-linearity to be able to obtain complex mappings from the input to the reservoir states (quantified by a kernel rank). When these requirements are fulfilled, it is possible to optimise a reservoir for a specific functionality[43, 44].

2.2.2 Implementations of Physical Reservoirs

Physical reservoirs can be implemented using a variety of methods. It is for example possible to use a bucket of water as the reservoir[45]. The broad requirements on reservoirs have led to different implementations. These approaches can be subdivided into two main categories. The first category uses nonlinear components to build up the complex system required for reservoir computing. In optical and optoelectronic systems, this can be achieved using a single nonlinear node with time-delayed feedback[46, 47, 48, 49, 50]. Reservoirs can be made by interconnecting the system in a complex structure[51, 52, 53] or by creating and using memristors with inherent fading memory[54]. In the second category, researchers exploit physical systems that inherently have the three properties described above to be used as natural reservoirs (reservoirs based on the inherent behaviour of matter)[37, 38]. By combining these natural reservoirs with the RC framework, it is possible to exploit the material's complex behaviour for computation without knowing all the internal details of the system[44, 55]. This approach is, for example, used to exploit the computational power of a network of cadmium-selenide quantum dots by making them sensitive to the surroundings pH, redox potential, or specific ion concentrations[56]. The method is also successfully implemented in organic networks of sulfonated polyaniline (SPAN)[57] and single-walled carbon nanotube/polymer reservoirs[15]. Using single-walled carbon nanotube/polymer as an example, it was shown that tuning the properties of a reservoir allows for the increase of the computational capabilities of the reservoir. For this approach, the reservoir is tuned following the evolution-in-materio approach.

2.2.3 Evolution-in-Materio

Evolution-in-materio is a method that uses artificial computer-controlled evolution of complex material systems. This method was developed to use unknown chemical and physical processes for computing applications[39]. The artificial evolution for these systems is performed using a genetic algorithm (GA). During training with a GA first an initial generation of genomes is defined. A genome is a set of tuneable parameters of the material system. This genome is, together with the input data,

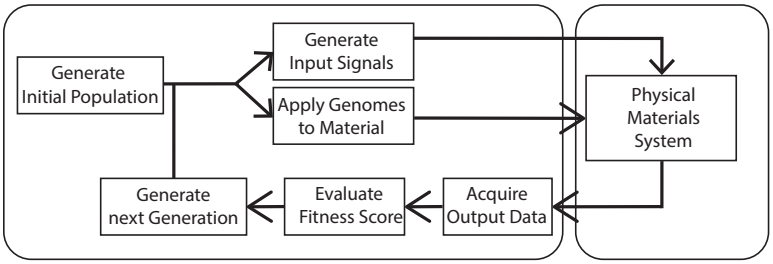


Figure 2.5: Schematic representation of a genetic algorithm with the digital computer domain on the left and the physical domain on the right[58].

applied to the configurable material. The output signal is ranked based on a so called fitness (similarity to the desired output). Based on this fitness ranking the next generation of genomes is created. This is done by keeping some of the old genomes, crossbreeding among genomes, mutation of genomes and by generating completely new genomes. This process is performed for a certain number of generations or until a specified fitness threshold is reached[58]. A schematic representation of a GA loop is shown in Figure 2.5. The trainability of a material using a GA makes this material a candidate for neuromorphic engineering since the materials adheres to the requirement of interaction with its surroundings as well as the capability to be taught a specific response to information, only not guaranteeing non-volatile memory.

2.2.4 Implementations of Evolution-in-Materio

One of the first materials used for evolution-in-materio are liquid crystals. Liquid crystals exist in a mesomorphic state or, in other words, a state that is in between a liquid and a solid crystal[59]. In these materials the orientation of the crystals is highly dependent on an externally applied electric field. Using this electric field it is possible to tune the electronic properties. Combining this tuneability with a GA, it is possible to train liquid crystals to perform the reconfigurable Boolean logic benchmark task[60]. This benchmark requires both tuneability as well as complex nonlinear behaviour. Another material system used for evolution-in-materio is a disordered network of gold nanoparticles (AuNPs)[14]. This network is made out of AuNPs with a diameter of 20 nm that are interconnected by insulating molecules (1-octanethiols). At low enough temperatures (below 5 K) the nanoparticles act as single electron transistors (SET)[61]. By using an external electric field (by applying voltages), it is possible to move the energy levels of the SETs, allowing the particles to switch between the single electron tunneling and Coulomb blockade regime[62]. This allows tuning of the current paths through the material, which has been used to train the device to perform the reconfigurable Boolean logic benchmark[14]. The devices under

investigation in this thesis are a follow-up from the AuNP networks. Dopant network processing units (DNPUs) originate from the idea that it is also possible to achieve single electron tunneling in a so called single-atom transistor where a single atom plays the role of a single nanoparticle[63].

2.3 Dopant Network Processing Units

DNPUs are made of a doped semiconductor. For the DNPUs used in this thesis, we generally use boron-doped silicon. Because the ionisation energy of the dopant atoms is higher than the charging energy of the AuNP quantum dots they function at higher temperatures, going from an operating temperature of 5 K to 160 K. Even room-temperature operation was achieved[16]. At these temperatures and with the correct doping concentration, the conduction in these devices is not solely governed by Coulomb blockade but also by variable range hopping (VRH). Hopping conduction occurs when the charges remain localised on impurities (dopants in the case of DNPUs). In this case, charges can only move from one impurity site to another through hopping. Since the sites do not have the same energy, the hopping process between two sites i and j requires a gain or loss of energy that is associated with the emission or absorption of a phonon. Figure 2.6a illustrates the hopping process. The hopping rate (Γ_{ij}) can be described using Equation 2.1[64].

$$\Gamma_{ij} = \begin{cases} \nu_0 \exp\left(\frac{-2r_{ij}}{a_b} - \frac{\epsilon_{ij}}{k_B T}\right) & , \epsilon_{ij} \geq 0 \\ \nu_0 \exp\left(\frac{-2r_{ij}}{a_b}\right) & , \epsilon_{ij} < 0 \end{cases} \tag{2.1}$$

were ν_0 is a hopping prefactor, r_{ij} is the distance between the two impurity sites, a_b represents the Bohr radius, ϵ_{ij} the difference between the energies of the impurity sites and $k_B T$ represents the thermal energy. When ϵ_{ij} is smaller than 0, the process generates a phonon. Similarly, when ϵ_{ij} is larger than 0, the energy of a phonon is required for the hop to occur.

Looking at Equation 2.1 there are two hopping conduction regimes, nearest neighbour-hopping (NNH) and variable range hopping (VRH). NNH occurs when the thermal energy is sufficiently high for the $\frac{\epsilon_{ij}}{k_b T}$ part the equation to become smaller than distance dependent side. In this case, hopping occurs between sites close together, hence between nearest-neighbour impurity sites. When the thermal energy is low, the second term will become relevant. This results in a trade-off between the distance and energy difference between the two impurity sites i and j. Due to this trade-off, hopping will occur over different ranges and VRH will govern the conduction[64]. Depending on the temperature, there are two different variable range hopping mechanisms, Mott VRH and Efros-Shklovskii VRH[65]. For Efros-Shklovskii VRH, the thermal energy becomes so small that the Coulomb gap starts playing a role in the conduction. This Coulomb gap originates from the long range interaction between localised electrons

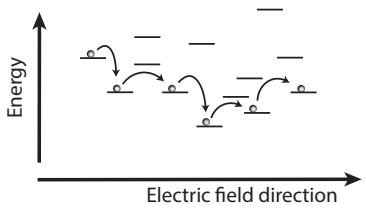


Figure 2.6: Schematic representation of a series of hopping events under the influence of an external electric field[67].

and results in a small gap in the density of states near the Fermi level[66]. Due to the differences in behaviour, the conductances of these hopping mechanisms have different temperature dependencies.

DNPUs exploit the tuneable nonlinear behavior of VRH conduction for evolution-in-materio. By using multiple electrodes to change the electric field, the energies of the impurity states are tuned. In this way the path charges will take through the disordered material is changed, which changes the resistance and thus the output current. This results in a highly tuneable nanoelectronic device that is capable of being trained to perform basic benchmark tasks. In this thesis we use DNPUs for classification tasks, get a first idea of the potential to connect these devices together to obtain more complex functionality, and look at combining evolution-in-materio with the RC approach to extract more complex nonlinear behaviour from DNPUs.

References

[1] G. Indiveri and T. Horiuchi. Frontiers in neuromorphic engineering. *Frontiers in Neuroscience*, 5, 2011. doi: 10.3389/fnins.2011.00118.

[2] K. J. Friston, C. D. Frith, R. J. Dolan, C. J. Price, S. Zeki, J. Ashburner, and W. Penny. *Human brain function*. Elsevier, 2003. ISBN 9780080472959.

[3] S. Furber. Large-scale neuromorphic computing systems. *Journal of Neural Engineering*, 13(5):051001, aug 2016. doi: 10.1088/1741-2560/13/5/051001.

[4] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice. The rise of intelligent matter. *Nature*, 594(7863):345–355, Jun 2021. doi: 10.1038/s41586-021-03453-y.

[5] G. Montavon, W. Samek, and K.R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018. ISSN 1051-2004. doi: 10.1016/j.dsp.2017.10.011.

[6] A. Nguembang Fadja, E. Lamma, and F. Riguzzi. Vision inspection with neural networks. 12 2018.

[7] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. doi: 10.1037/h0042519.

[8] W. Zhou. Verification of the nonparametric characteristics of backpropagation neural networks for image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 37(2):771–779, 1999. doi: 10.1109/36.752193.

[9] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv*, 2018. doi: 10.48550/arXiv.1803.08375.

[10] N. Kanwar, A. K. Goswami, and S. P. Mishra. Design issues in artificial neural network (ann). In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4, 2019. doi: 10.1109/IoT-SIU.2019.8777337.

[11] P. P. Jagtap. Embedded systems architecture-review. *IRJET*, 5(11):227–229, 2018.

[12] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.

[13] R. Islam, H. Li, P. Chen, W. Wan, H-Y. Chen, B. Gao, H. Wu, S. Yu, K.a Saraswat, and H-S. P. Wong. Device and materials requirements for neuromorphic computing. 52(11):113001, jan 2019. doi: 10.1088/1361-6463/aaf784.

[14] S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature Nanotechnology*, 10(12):1048–1052, Dec 2015. doi: 10.1038/nnano.2015.207.

[15] M. Dale, J. Miller, S. Stepney, and M. Trefzer. Evolving carbon nanotube reservoir computers. volume 9726, pages 49–61, 07 2016. doi: 10.1007/978-3-319-41312-9_5.

[16] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.C. Ruiz Euler, H. Broersma, P.r A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel. Classification with a disordered dopant-atom network in silicon. *Nature*, 577(7790):341–345, Jan 2020. doi: 10.1038/s41586-019-1901-0.

[17] Q. Xia and J. J. Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18(4):309–323, Apr 2019. doi: 10.1038/s41563-019-0291-x.

[18] H. T. Peng, M. A. Nahmias, T. F. de Lima, A. N. Tait, and B. J. Shastri. Neuromorphic photonic integrated circuits. *IEEE Journal of Selected Topics in Quantum Electronics*, 24(6):1–15, 2018. doi: 10.1109/JSTQE.2018.2840448.

[19] S. Kim, C. Du, P. Sheridan, W. Ma, S. Choi, and W. D. Lu. Experimental demonstration of a second-order memristor and its ability to biorealistically implement synaptic plasticity. *Nano Letters*, 15(3):2203–2211, Mar 2015. doi: 10.1021/acs.nanolett.5b00697.

[20] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou. Stochastic phase-change neurons. *Nature Nanotechnology*, 11(8):693–699, Aug 2016. doi: 10.1038/nnano.2016.70.

[21] Daniele G. *A variability study of PCM and OxRAM technologies for use as synapses in neuromorphic systems.* PhD thesis, Micro and nanotechnologies/Microelectronics. Université Grenoble Alpes, 2015.

[22] D. Kwon, K. M. Kim, J. H. Jang, and Jeon et al. Atomic structure of conducting nanofilaments in tio_2 resistive switching memory. *Nature Nanotechnology*, 5(2):148–153, Feb 2010. doi: 10.1038/nnano.2009.456.

[23] T. Breuer, L. Nielen, B. Roesgen, R. Waser, V. Rana, and E. Linn. Realization of minimum and maximum gate function in ta_2o_5-based memristive devices. *Scientific Reports*, 6(1):23967, Apr 2016. doi: 10.1038/srep23967.

[24] M. M. Shulaker, G. Hills, R. S. Park, and R. T. Howe et al. Three-dimensional integration of nanotechnologies for computing and data storage on a single chip. *Nature*, 547(7661):74–78, Jul 2017. doi: 10.1038/nature22994.

[25] A. Wedig, M. Luebben, D-Y. Cho, M. Moors, K. Skaja, V. Rana, T. Hasegawa, K. K. Adepalli, B. Yildiz, R. Waser, and I. Valov. Nanoscale cation motion in tao_x, hfo_x and tio_x memristive systems. *Nature Nanotechnology*, 11(1):67–74, Jan 2016. doi: 10.1038/nnano.2015.221.

[26] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646, Jan 2020. doi: 10.1038/s41586-020-1942-4.

[27] A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. BrightSky, and E. Eleftheriou. Tutorial: Brain-inspired computing using phase-change memory devices. *Journal of Applied Physics*, 124(11):111101, 2018. doi: 10.1063/1.5042413.

[28] D-H. Lim, S. Wu, R. Zhao, J-H. Lee, H. Jeong, and L.P. Shi. Spontaneous sparse learning for pcm-based memristor neural networks. *Nature Communications*, 12 (1):319, Jan 2021. doi: 10.1038/s41467-020-20519-z.

[29] N. K. Upadhyay, H. Jiang, Z. Wang, S. Asapu, Q. Xia, and J. Yang. Emerging memory devices for neuromorphic computing. *Advanced Materials Technologies*, 4(4):1800589, 2019. doi: 10.1002/admt.201800589.

[30] T. J. Kippenberg, R. Holzwarth, and S. A. Diddams. Microresonator-based optical frequency combs. *Science*, 332(6029):555–559, 2011. doi: 10.1126/science.1193968.

[31] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja, J. Liu, C. D. Wright, A. Sebastian, T. J. Kippenberg, W. H. P. Pernice, and H. Bhaskaran. Parallel convolutional processing using an integrated photonic tensor core. *Nature*, 589(7840): 52–58, Jan 2021. doi: 10.1038/s41586-020-03070-1.

[32] X. Xu, M. Tan, B. Corcoran, J. Wu, T. G. Nguyen, A. Boes, S. T. Chu, B. E. Little, R. Morandotti, A. Mitchell, D. G. Hicks, and D. J. Moss. Photonic perceptron based on a kerr microcomb for high-speed, scalable, optical neural networks. *Laser & Photonics Reviews*, 14(10):2000070, 2020. doi: https://doi.org/10.1002/lpor.202000070.

[33] C. Rios, M. Stegmaier, P. Hosseini, and D. Wang et al. Integrated all-photonic non-volatile multi-level memory. *Nature Photonics*, 9(11):725–732, Nov 2015. doi: 10.1038/nphoton.2015.182.

[34] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature*, 569(7755):208–214, May 2019. doi: 10.1038/s41586-019-1157-8.

[35] P.l A. Morton, J. B. Khurgin, and M. J. Morton. All-optical linearized mach-zehnder modulator. *Opt. Express*, 29(23):37302–37313, Nov 2021. doi: 10.1364/OE.438519.

[36] X. Xu, M. Tan, B. Corcoran, J. Wu, A. Boes, T. G. Nguyen, S. T. Chu, B. E. Little, D. G. Hicks, R. Morandotti, A. Mitchell, and D. J. Moss. 11 tops photonic convolutional accelerator for optical neural networks. *Nature*, 589(7840):44–51, Jan 2021. doi: 10.1038/s41586-020-03063-0.

[37] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019. doi: 10.1016/j.neunet.2019.03.005.

[38] K. Nakajima. Physical reservoir computing—an introductory perspective. *Japanese Journal of Applied Physics*, 59(6):060501, may 2020. doi: 10.35848/1347-4065/ab8d4f.

[39] J. F Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, pages 167–176. IEEE, 2002. doi: 10.1109/EH.2002.1029882.

[40] A. Sharkawy. Principle of neural network and its main types: Review. *Journal of Advances in Applied amp;amp; Computational Mathematics*, 7:8–19, Aug. 2020. doi: 10.15377/2409-5761.2020.07.2.

[41] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009. doi: https://doi.org/10.1016/j.cosrev.2009.03.005.

[42] H. Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. *GermanNational Research Center for Information Technology*, 2002.

[43] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. *Reservoir Computing in Material Substrates*, pages 141–166. Springer Singapore, Singapore, 2021. doi: 10.1007/978-981-13-1687-6_7.

[44] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. A substrate-independent framework to characterize reservoir computers. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2226):20180723, 2019. doi: 10.1098/rspa.2018.0723.

[45] C. Fernando and S." Sojakka. Pattern recognition in a bucket. In *dvances in Artificial Life*, pages 588–597. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-39432-7.

[46] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2 (1):468, Sep 2011. doi: 10.1038/ncomms1476.

[47] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Opt. Express*, 20(3): 3241–3249, Jan 2012. doi: 10.1364/OE.20.003241.

[48] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar. Optoelectronic reservoir computing. *Scientific Reports*, 2(1):287, Feb 2012. doi: 10.1038/srep00287.

[49] M. C. Soriano, S. Ortín, D. Brunner, L. Larger, C. R. Mirasso, I. Fischer, and L. Pesquera. Optoelectronic reservoir computing: tackling noise-induced performance degradation. *Opt. Express*, 21(1):12–20, Jan 2013. doi: 10.1364/OE.21.000012.

[50] S. Ortín, M. C. Soriano, L. Pesquera, D. Brunner, D. San-Martín, I. Fischer, C. R. Mirasso, and J. M. Gutiérrez. A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron. *Scientific Reports*, 5(1):14945, Oct 2015. doi: 10.1038/srep14945.

[51] S. Wolfram. *Cellular automata and complexity: collected papers*. crc Press, 2018. ISBN 0-201-62716-7.

[52] J. C. Coulombe, M. C. A. York, and J. Sylvestre. Computing with networks of nonlinear mechanical oscillators. *PloS one*, 12(6):e0178663, 2017. doi: 10.1371/journal.pone.0178663.

[53] G. Tanaka, R. Nakane, T. Yamane, and S. Takeda. Waveform classification by memristive reservoir computing. In *Neural Information Processing*, pages 457–465. Springer International Publishing, 2017. doi: 10.1007/978-3-319-70093-9_48.

[54] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu. Reservoir computing using dynamic memristors for temporal information processing. *Nature Communications*, 8(1):2204, Dec 2017. doi: 10.1038/s41467-017-02337-y.

[55] H. Sillin, R. Aguilera, H.-H. Shieh, A. Avizienis, M. Aono, A. Stieg, and J. Gimzewski. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology*, 24(38):384004, sep 2013. doi: 10.1088/0957-4484/24/38/384004.

[56] O. Obst, A. Trinchi, S. G. Hardin, M. Chadwick, I. Cole, T. H. Muster, N. Hoschke, D. Ostry, D. Price, K. N. Pham, and T. Wark. Nano-scale reservoir computing. *Nano Communication Networks*, 4(4):189–196, 2013. doi: https://doi.org/10.1016/j.nancom.2013.08.005.

[57] Y. Usami, B. van de Ven, D. G. Mathew, T. Chen, T. Kotooka, Y. Kawashima, Y. Tanaka, Y. Otsuka, H. Ohoyama, H. Tamukoh, H. Tanaka, W. G. van der Wiel, and T. Matsumoto. In-materio reservoir computing in a sulfonated

polyaniline network. *Advanced Materials*, 33(48):2102688, 2021. doi: https://doi.org/10.1002/adma.202102688.

[58] J. F. Miller, Simon L. H., and Gunnar T. Evolution-in-materio: Evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67, 2014. doi: 10.1007/s12065-014-0106-6.

[59] D Demus, J Goodby, GW Gray, HW Spiess, and V Vill. Handbook of liquid crystals, 1998.

[60] S. Harding and J. F Miller. Evolution in materio: Evolving logic gates in liquid crystal. In *Proc. Eur. Conf. Artif. Life (ECAL 2005), Workshop on Unconventional Computing: From cellular automata to wetware*, pages 133–149. Beckington, UK, 2005. doi: 10.1007/978-0-387-30440-3_190.

[61] K.K. Likharev. Single-electron devices and their applications. *Proceedings of the IEEE*, 87(4):606–632, 1999. doi: 10.1109/5.752518.

[62] R. van Damme, H. Broersma, J. Mikhal, C. Lawrence, and W. G. van der Wiel. A simulation tool for evolving functionalities in disordered nanoparticle networks. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 5238–5245, 2016. doi: 10.1109/CEC.2016.7748354.

[63] M. Fuechsle, J. A. Miwa, S. Mahapatra, H. Ryu, S. Lee, O. Warschkow, L. C. L. Hollenberg, G. Klimeck, and M. Y. Simmons. A single-atom transistor. *Nature Nanotechnology*, 7(4):242–246, Apr 2012. doi: 10.1038/nnano.2012.21.

[64] A. J. Miller and E. Abrahams. Impurity conduction at low concentrations. *Physical Review*, 120:745–755, 1960. doi: 10.1103/PhysRev.120.745.

[65] A. L. Efros and B. I. Shklovskii. Coulomb gap and low temperature conductivity of disordered systems. 8(4):L49–L51, feb 1975.

[66] A. L. Efros. Coulomb gap in disordered systems. *Journal of Physics C: Solid State Physics*, 9(11):2021–2030, jun 1976. doi: 10.1088/0022-3719/9/11/012.

[67] J. van Gelder. *Using lowly doped silicon to create reconfigurable Boolean logic gates using an evolutionary algorithm.* PhD thesis, University of Twente, 2017.

3

Fabrication and Characterisation of Dopant Network Processing Units

In this chapter, we focus on the fabrication of dopant network processing units (DNPUs) performed in the cleanroom of MESA+. First, we describe the main fabrication processes. After this, we analyse which aspects of fabrication limit the yield and propose potential methods of mitigating these aspects. Finally, we elaborate on the electronic characterisation by describing the general set-up used to perform the experiments presented in this thesis.

3.1 Fabrication of Dopant Network Processing Units

The fabrication of dopant network processing units (DNPUs) is based on silicon technologies[1, 2] and combines micron-scale fabrication techniques, performed on 4-inch wafers, with nano-scale fabrication techniques, performed on 1x1 cm^2 chips. The wafer-scale processing is centered around photolithography, which allows us to quickly fabricate the repeating structures required for all DNPUs. At the scale of a single chip, the processing is based on the use of electron-beam lithography (EBL) since this method allows for the fabrication of nano-scale features. Another advantage of EBL is the high degree of flexibility allowing us to easily change the design for each fabrication round[3].

In principle, DNPUs can be fabricated using a wide variety of host and dopant combinations. The main requirement is that the device is operated in the variable range hopping (VRH) regime such that the conduction through the doped material becomes nonlinearly dependent on an applied electric field (see Chapter 4[4]).

In this chapter, the fabrication of two types of DNPUs, boron-doped silicon and arsenic-doped silicon, is explained. The main focus will be on boron DNPUs since such DNPUs are mostly used in the rest of the work described in this thesis. The differences between the boron-doped and arsenic-doped DNPUs are indicated to give an idea about the small variations that need to be considered when fabricating DNPUs with different host-dopant combinations.

3.2 Micron-scale Fabrication

DNPUs can be fabricated using a variety of silicon wafers, from intrinsic to lightly p- and n-type doped silicon. Choosing a wafer with a background doping concentration lower than the concentration required for VRH will make the fabrication easier by reducing the implantation depth due to the reduced space in the silicon. For boron

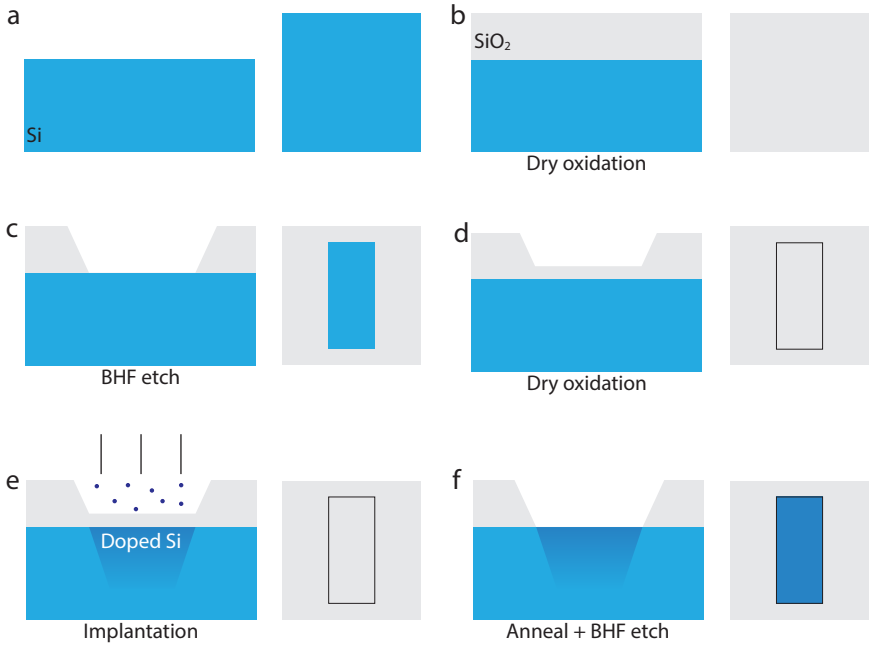


Figure 3.1: Schematics of the side- and top view during micron-scale fabrication. a, Initial silicon wafer (light blue). b, The wafer after dry oxidation to grow 300 nm SiO_2 (light grey). c, Wafer after BHF etching the SiO_2 window. d, Growth of 35 nm (25 nm for As) SiO_2 used to create the desired dopant profile in the Si. e, Ion beam implantation of the dopants. f, Annealing to incorporate the dopants into the Si lattice and a BHF etch to expose the highly doped silicon surface (dark blue).

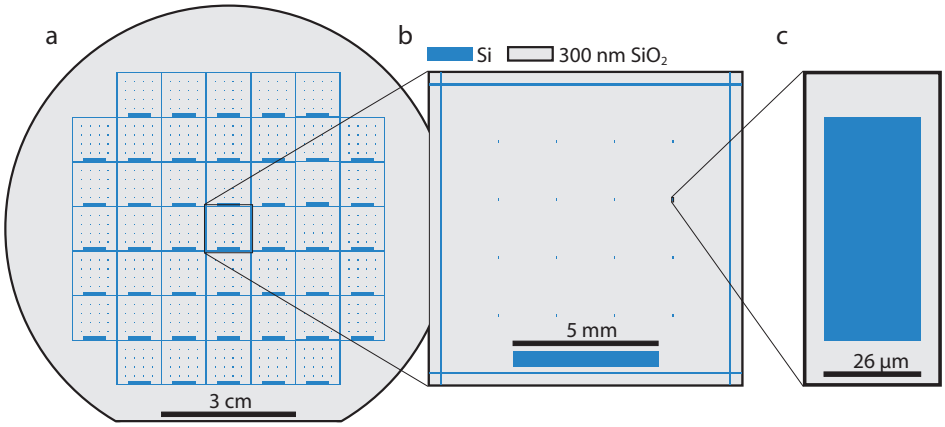


Figure 3.2: Schematic of the wafer after micron-scale fabrication. a, The 4-inch wafer. b, Zoom in on the region of a single 1x1 cm^2 chip. c, Zoom in on a single 26 x 60 μm^2 implanted region on which the DNPU will be fabricated.

doped DNPUs, we use a one-side polished n-type Si(100) wafer (resistivity, $\rho = 1.00 - 10.00\ \Omega$cm). Such a wafer has a background doping between $1.3 \cdot 10^{15}$ and $1.5 \cdot 10^{16}\ cm^{-3}$ which, is below the desired DNPU dopant concentration of approximately $5 \cdot 10^{17}\ cm^{-3}$ (see Chapter 4[4]) such that this background doping does not govern the conductance while still limiting the dopant implantation depth forcing the desired concentration closer to the electrodes. For arsenic doping, a one-side polished p-type Si(100) wafer ($\rho = 5.00 - 10.00\ \Omega$cm) is used. A schematic of the cross-section of the micron-scale fabrication steps is shown in Figure 3.1. A detailed process flow can be found in Section 3.5.1.

3.2.1 Defining the Oxide Mask

To define the regions of the wafer where the dopants will be implanted, we use a silicon oxide (SiO_2) mask. To create this mask, an oxide layer needs to be grown. Before dry oxidation, the wafer must be thoroughly cleaned. Potential organic residue is removed using nitric acid (HNO_3) after which the thin layer of native oxide still present is removed using 1% hydrofluoric acid (HF). After cleaning, the wafer is inserted in an ultra-clean furnace where it is placed with 10 dummy wafers in front and behind it. These dummy wafers ensure a laminar flow in the furnace tube to obtain a homogeneous oxide growth. After the wafers are placed in the oven, the temperature is ramped to 1100 °C (5 °C/min) at which the oxidation is performed in an O_2 atmosphere. Using this process, 300 nm SiO_2 is grown in 4.5 hours (see Figure 3.1b).

Photolithography is used to define the implantation regions in this silicon oxide. For this process, we start by spinning a monolayer of HexaMethylDiSilazane (HMDS). On top of this primer, we spin Olin OiR 907-17 as the photoresist. After exposure and development, we etch the SiO_2 in the now exposed regions using buffered HF (BHF) (see Figure 3.1c). As the final step before implantation, dry oxidation is used to grow 35 nm SiO_2 (25 nm for arsenic) at 1050 °C for 14 minutes (8 minutes for arsenic) to realise the highest implantation concentration at the surface (see Figure 3.1d). Figure 3.3 shows the simulated implantation profiles in the silicon (using an online simulator [5]), where the green coloured box indicates the 35 nm (25 nm) SiO_2. It can be observed that the dopant concentration at the surface exceeds 10^{19} cm^3, which is needed to achieve an ohmic contact with the metal electrodes[6].

3.2.2 Ion Implantation

The ion implantation is performed by Ion Beam Services (IBS)[7] where the wafer is implanted at 9 KeV with $3.5 \cdot 10^{14}$ atoms/cm^3 (35 keV with 10^{14} atoms/cm^3 for arsenic) (see Figure 3.1e). After implantation, we perform rapid thermal annealing (RTA) to incorporate the dopants in the silicon lattice[8]. This is performed at 1050 °C for 7 seconds. After RTA the thin oxide layer is removed by etching with BHF to expose the highly doped silicon surface (see Figure 3.1f).

The final step before dicing is the fabrication of the EBL markers. These markers are 20 x 20 μm^2 squares of 1 nm titanium (Ti) and 50 nm platinum (Pt). First, the markers are defined using photolithography, using the same resist as above. The markers are finally fabricated using a combination of sputtering and lift-off[9]. An illustration of the final wafer (without the EBL markers) is shown in Figure 3.2 where the light grey areas are the 300 nm SiO_2 and the blue areas the doped silicon. The blue lines that define the 1x1 cm^2 indicate where the wafer will be diced. The 5 mm wide regions are there to allow for further investigation of the doping profile. The rectangle shown in Figure 3.2c is the implantation region in which the active region of the DNPU is contacted using nano-scale electrodes.

3.3 Nanoscale Fabrication

When all micron-scale structures are fabricated, and the wafer is diced we are left with 45 chips each having 16 implantation windows. During nano-scale processing, metal electrodes are fabricated on top of the silicon surface of one of the chips, to form ohmic contacts with the highly doped silicon. Next, the silicon is etched to reach the desired doping concentration at which VRH is the governing conduction mechanism. A schematic side view of the nano-scale fabrication is shown in Figure 3.4. A detailed process flow can be found in Section 3.5.2.

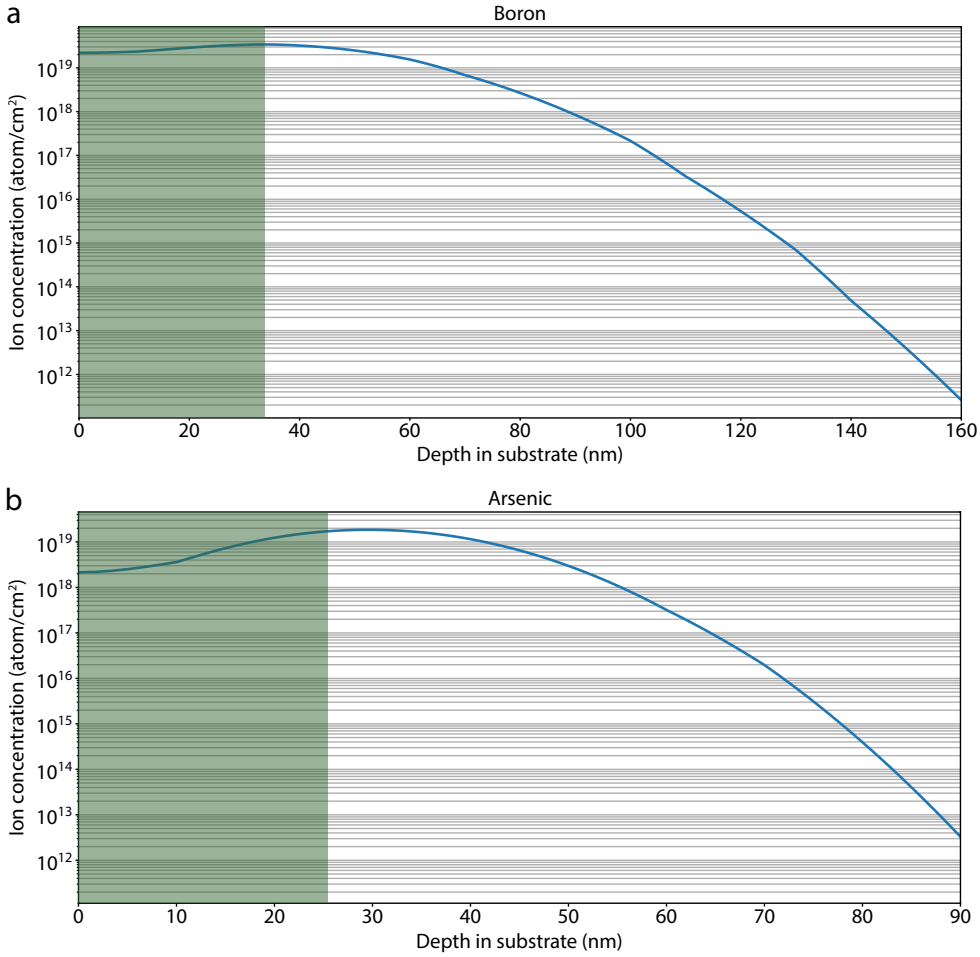


Figure 3.3: Simulation of the ion concentration profile. a, The ion concentration profile when using boron dopants. The green box indicates the part of the dopants trapped by the thin SiO_2 layer. The right edge of this box is at the surface of the silicon. b, Same as in a when using arsenic dopants[5].

3.3.1 Electrode Fabrication

To contact the DNPU with metal electrodes, we use a combination of EBL, electron-beam evaporation, and lift-off[9]. First, the chips are cleaned using acetone and isopropyl alcohol (IPA). Next, a layer of 175 nm polymethyl methacrylate (PMMA) is spin-coated to function as the EBL resist. During EBL we expose the PMMA in the regions where we want to have electrodes. The DNPUs in this thesis either have 8 or

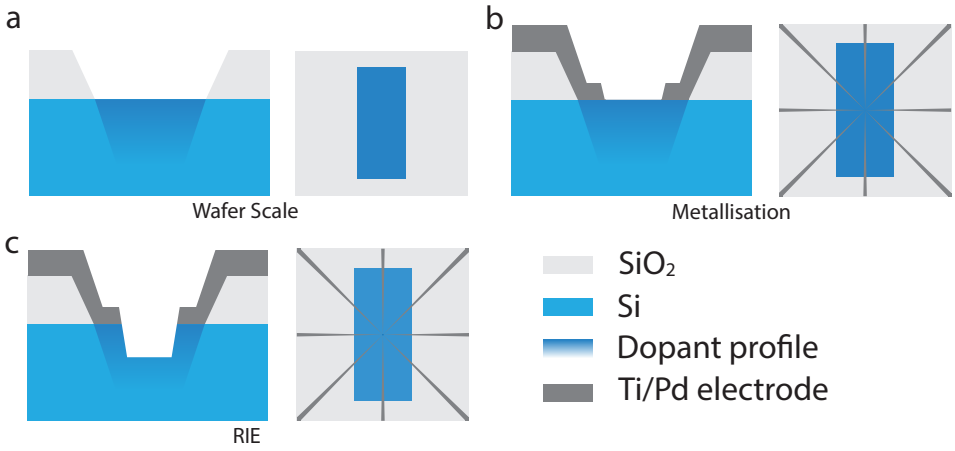


Figure 3.4: Schematics of the side- and top- view during the nanoscale fabrication. a, The device after the wafer-scale fabrication, with the Si in blue, the SiO_2 in light grey, and the doping concentration represented by different shades of dark blue. b, The device after EBL, e-beam evaporation, and lift-off to fabricate the metal electrodes (dark grey). c, The device after RIE to the desired doping concentration.

12 electrodes equally spaced on a circle with a diameter of 300 nm (see Figure 3.5 for an illustration of the final chip with 8 electrodes). Here the diameter of the circle and resolution of EBL limits the number of electrodes. Since the width or our electrodes is approximately 50 nm, for the 300 nm diameter used, we are limited to approximately 18-20 electrodes[10]. However, we expect that, when placing the electrodes this close together, the influence of individual electrodes will be difficult to distinguish from one another. An electric field similar to the one created by, for example, 3 electrodes close together, could also be created using a single electrode. After e-beam exposure, the PMMA is developed in a methyl isobutyl ketone:IPA (MIBK:IPA) solution (1:3) for 45 seconds after which it is rinsed with IPA.

After defining the electrode positions in the PMMA, we need to deposit the metal on top of the chip. However, we first have to clean the exposed silicon surface using UV-ozone for 5 mins, to remove potential organic residue, after which we etch the native SiO_2 using 1% HF for 10 s to ensure good contact between the metal and the highly doped silicon. Here it is important not to etch the SiO_2 for too long such that the underetch does not widen the pattern. Now that the chip is cleaned, the metal is deposited using e-beam evaporation. First, 1.5 nm of titanium is deposited as a sticking layer, after which 25 nm palladium is evaporated to form a metal layer on top of the silicon/PMMA (25 nm aluminium for arsenic due different requirements for n- and p- type doping). To remove the excess metal on top of the PMMA we use lift-off. This is done by heating dimethyl sulfoxide (DMSO) to 90 °C in which we

suspend the chip for $15 - 30$ mins. When the metal layer starts to form cracks, we perform gentle ultrasonication in a bath at 80 °C, in batches of 1 minute, until the excess metal has been completely lifted off. After this, we end up with the structure illustrated in Figure 3.4b.

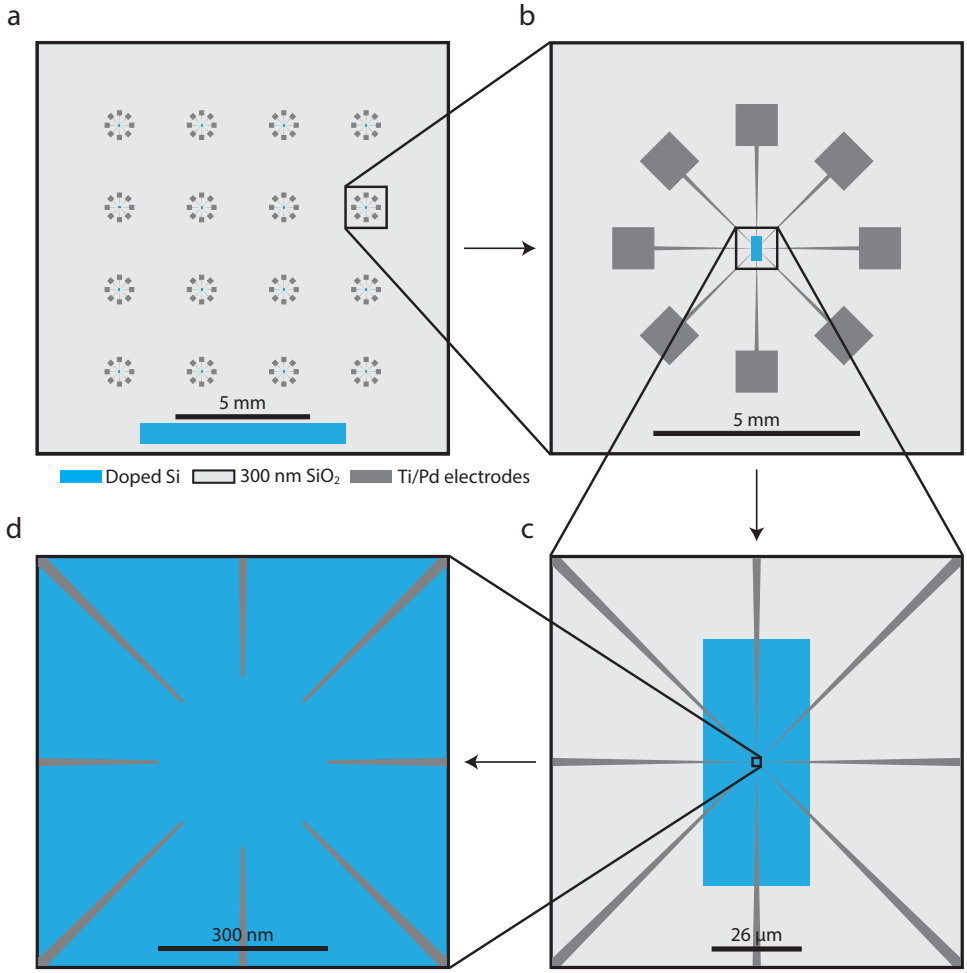


Figure 3.5: Schematic of the chip after nano-scale fabrication. a, Full 1x1 cm^2 chip with 16 DNPUs. Blue represents the doped silicon area, the areas in light grey represent the SiO_2 and the dark grey structures represent the metal electrodes. b, single device with 8 electrodes. c, The doped 26 x 60 μm^2 area. d, The active area with the electrodes evenly spaced in a circle with a 300 nm diameter.

3.3.2 Reactive Ion Etching

After fabricating the metal electrodes on top of the highly doped silicon, we need to ensure that the correct doping concentration for variable range hopping is achieved. Using Figure 3.3 we can see that, to reach a concentration of $5 \cdot 10^{17}$ cm^{-3}, we need to remove approximately 70 nm for boron and approximately 30 nm for arsenic DNPUs.

We remove the excess layer of doped silicon using reactive ion etching (RIE)[11]. This step is performed in a gas flow of fluoroform (CHF_3) at 25 cm^3/min and oxygen (O_2) at 5 cm^3/min while using an energy of 25 W at the back electrode for $3 - 7$ minutes ($1 - 3$ minutes for arsenic). Under these conditions, the metal electrodes are not etched away and directly function as a mask to keep them in contact with the highly doped silicon. A side schematic of the final chip can be seen in Figure 3.4c and a top schematic in Figure 3.5.

The yield of DNPUs is mostly limited by this etching step. The $3 - 7$ minutes is too big a variation to reliably fabricate DNPUs and is limited by the instability in RIE when attempting to perform such a shallow etch. In combination with the small region in which the conduction of the DNPU is governed by VRH while still being conductive enough to measure (10 nm), this indicates that it is important to improve the RIE fabrication process. Generally, we solved this issue by fabricating many chips and performing trial and error to discover how long the etch time should be during a given session. When comparing boron doping with arsenic doping, we observe that the need to etch less deep reduces the variability in time. This is attributed to the reduced maximum time that is needed due to the arsenic dopants being closer to the surface. Using heavier dopants might increase this depth reduction, making fabrication more reliable. Another method to circumvent the problem is by performing two implantation steps with different parameters in an attempt to widen the etch depth range in which the desired dopant concentration is achieved, increasing the region in which DNPUs operate as desired. It must be noted, however, that besides this RIE step the fabrication of DNPUs is relatively reliable. On average, when the correct concentration is reached on a chip, more than half of the 16 DNPUs show the desired behaviour. The other DNPUs tend to have broken electrodes caused by the lift-off process.

3.4 Characterisation of Dopant Network Processing Units

A schematic of the measurement setup for the electronic characterisation of DNPUs is shown in Figure 3.6. DNPUs operate in the variable range hopping (VRH) regime. For VRH, besides the correct dopant concentration achieved during fabrication, reaching the correct temperature is crucial. For boron-based DNPUs, this is in the range of 70 K to 160 K, see Chapter 4[4]. For convenience, we operate DNPUs at 77 K or liquid nitrogen temperature. To put the DNPU in liquid nitrogen, we connect it to a printed circuit board (PCB) that is attached to a dipstick. The electronic connections

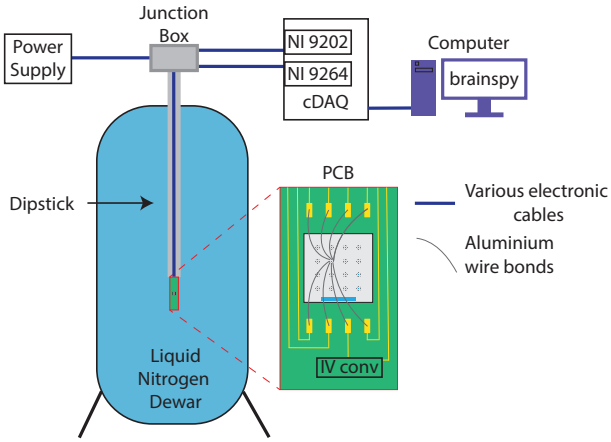


Figure 3.6: Schematic of the measurement setup used to characterise the dopant network processing units at 77K.

on this PCB are wired to a junction box that regulates the connections to external electronics. To connect the PCB and junctionbox we use either flat or coax cables. Flat cables allow us to have many wires in the limited tube size of the dipstick. Coax cables reduce the cross-talk between the cables, improving bandwidth and reducing noise. The PCB, device and cabling are inside the metal housing of the dipstick, which shields the electronics inside. Using such a dipstick, the DNPU is inserted in a dewar with liquid nitrogen, see the illustrated liquid nitrogen dewar in Figure 3.6.

Based on the combination of voltages applied a DNPU will output a specific current. Since the currents are often in the nA range, we use a low-temperature IV-converter to convert the small currents into voltage signals[12]. We want this IV-converter to be close to the device to make the signal in the wires less susceptible to cross-talk and external noise sources. The ratio of current to voltage conversion can be tuned by the feedback resistance of the IV converter. We have used different feedback resistances to either reduce noise (100 MΩ feedback resistance) or increase the output range (10 MΩ feedback resistance). This consideration is crucial since the IV-converter is the main factor for reducing noise during slow measurements (1 kHz sampling or less). However, an increased output range increases the chance of finding the desired DNPU behaviour. Since for most experiments, reducing noise is crucial, we mostly used the 100 MΩ feedback resistance. The IV-converter combines this feedback resistance with an operational amplifier (op-amp). For this op-amp to function we need to supply it with 5 V and -5 V. These voltages are provided using the external power supply connected to the PCB in the junction box as illustrated in Figure 3.6.

To precisely control the behaviour of a DNPU, we need accurate control over the supplied voltages. This is achieved using a digital to analogue converter (DAC). For the experiments presented in this thesis, we generate the voltages using a NI 9264 DAC module. This module is combined with a NI 9202 analogue to digital converter (ADC) module to read out the voltages generated by the DNPU + IV converter. These modules are placed in a NI compact data acquisition chassis (cDAQ). Using a modular system allows us to adapt the number of modules and offers flexibility in changing modules when required. This cDAQ chassis with the modules is controlled using the SkyNEt and brainspy python code (`https://github.com/BraiNEdarwin`) that we have developed. This code utilises the nidaqmx python package[13] for its cDAQ drivers.

To connect all of this to the DNPU, we use a specially designed PCB. The PCB has DAC channels that go from the PCB via the junction box to the NI 9264 DAC module and ADC channels first go via the low-temperature IV converter to the junction box and finally to the NI 9202 ADC module. We attach the DNPU to the PCB using double-sided tape. The DNPU electrodes are connected to the PCB electrodes using a wire bonder. This wire bonder uses an aluminium wire, with a diameter of 25 μm, to wire the PCB electrodes to the DNPU electrodes, as illustrated in the PCB zoom-in of Figure 3.6. The created connections will determine the input and output electrodes during the sample preparation stage.

Based on the general design presented above, we have made three additional PCB/ dipstick combinations. The first combination uses flat cables to create a PCB with space for five devices having 35 DAC and 5 ADC channels. The second dipstick + PCB is used for experiments that require flexibility in the number of input and output channels. This set-up has space for one DNPU, of which the electrodes can be wire bonded to 12 DAC and 12 ADC channels. The third set-up is used to test multiple devices in one cooling cycle (insert of the dipstick in the liquid nitrogen). We designed a PCB with space for eight devices. This PCB has 8 DAC and 1 ADC channels and utilises electronic switches to define the device to which these channels are connected. Using the above-described set-ups/PCB designs, we have performed experiments on DNPUs. We present the results of the experiments in the following chapters.

3.5 Appendix: Process Flow

3.5.1 Micron-scale Process Flow

Step 1: The substrate

DNPU type	boron	arsenic
Wafer	Si <100>	Si <100>
Background dopant	N/Phosphorus	P/Boron
Resistivity	$1.00 - 10.00\ \Omega$cm	$5.00 - 10.00\ \Omega$cm

Step 2: Dry Oxidation 1

Process	Cleaning in 99% HNO_3 (WB 14)
Temperature	Room Temperature
Time	5min in beaker 1, 5 min in beaker 2
Process	Quick Dump Rinse (Clean using DI water)
Process	Cleaning in 69% HNO_3 (WB 14)
Temperature	95 °C
Time	10 min
Process	Quick Dump Rinse (Clean using in DI water)
Process	Etching in 1% HF (WB 15)
Temperature	Room Temperature
Time	1 min
Process	Quick Dump Rinse (Clean using DI water)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec
Process	Dry Oxidation of Silicon (H1 furnace)
Temperature	1100 °C
O_2 flow	5 L/min
Pressure	Standard Atmosphere
Ramp	5 °C/min
Time	4.5 hours
Target thickness	300 nm

Step 3: Optical Lithography 1, defining the doped regions.

Process	Dehydration Bake (on a hotplate)
Temperature	120 °C
Time	5 min
Process	Spin Coating HMDS (WB 21)
Spin speed	4000 rpm
Time	30 sec
Process	Spin Coating Olin OiR 907-17
Spin speed	4000 rpm
Time	30 sec
Process	Prebake (on a hotplate)
Temperature	95 °C
Time	90 sec
Process	Resist exposure (Use the mask design from Figure 3.2a)
Separation	50 μm
Contact Mode	Soft Contact
Exposure Time	4.2 sec
Process	Postbake (on a hotplate)
Temperature	120 °C
Time	60 sec
Process	Resist Development (WB 21)
Time	30 sec beaker 1, 15-30 sec beaker 2
Process	Quick Dump Rinse (Clean using DI water)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec
Process	Postbake (on a hotplate)
Temperature	120 °C
Time	10 min

Step 4: BHF Etching 1

Process	Etching in Buffered HF (1:7) (WB 12)
Temperature	Room temperature
Etch Rate	60-80 nm/min
Time	5 min
Process	Quick Dump Rinse (Clean using DI water)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec

Step 5: Dry Oxidation 2

Process	Cleaning in 99% HNO_3 (WB 14)
Temperature	Room Temperature
Time	10min
Process	Quick Dump Rinse (Clean using DI water)
Process	Cleaning in 69% HNO_3 (WB 14)
Temperature	95 °C
Time	10 min
Process	Quick Dump Rinse (Clean using DI water)
Process	Etching in 1% HF (WB 15)
Temperature	Room Temperature
Time	1 min
Process	Quick Dump Rinse (Clean using DI water)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec
Process	Dry Oxidation of Silicon (H1 furnace)
Temperature	1050 °C
O_2 flow	5 L/min
Pressure	Standard Atmosphere
Ramp	5 °C/min
Time	14 min for Boron and 9 min for Arsenic
Target thickness	35 nm for Boron and 25 nm for Arsenic

Step 6: Ion Implantation

Process	Ion implantation (Ion Beam Services)
Species	Boron or Arsenic
Energy	10 kEV for Boron and 25 kEV for Arsenic
Dose	$5 * 10^{15}$ atoms/cm^2 for Boron and $1 * 10^{14}$ atoms/cm^2 for Arsenic
Process	Rapid Thermal Annealing
Temperature	1050 $°C$
Time	7 sec

Step 7: BHF etching 2

Process	Etching in Buffered HF (1:7) (WB 12)
Temperature	Room temperature
Etch Rate	60-80 nm/min
Time	1 min
Process	Quick Dump Rinse (Clean using DI water)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec

Step 8: Optical Lithography 2, defining wafer-scale EBL markers.

Process	Dehydration Bake (on a hotplate)
Temperature	120 °C
Time	5 min
Process	Clean using DI wSpin Coating HMDS (WB 21)ater
Spin speed	4000 rpm
Time	30 sec
Process	Spin Coating Olin OiR 907-17
Spin speed	4000 rpm
Time	30 sec

Process	Prebake (on a hotplate)
Temperature	95 °C
Time	90 sec
Process	Resist exposure (Use the mask design from Figure 3.2a)
Separation	50 μm
Contact Mode	Soft Contact
Exposure Time	4.2 sec
Process	Postbake (on a hotplate)
Temperature	120 °C
Time	60 sec
Process	Resist Development (WB 12)
Time	45-60 sec
Process	Quick Dump Rinse (Clean using DI water)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec
Process	Postbake (on a hotplate)
Temperature	120 °C
Time	10 min

Step 9: Sputtering and Lift-off 1

Process	Sputtering (T'COathy)
Material 1	Titanium
Thickness	5 nm
Materials 2	Platinum
Thickness	45 nm
Process	Lift-off (WB 11)
Chemical	Acetone
Time	10 min (or longer if the metal is not completely lifted-off)
Post Treatment	Rinse with Acetone (30 sec) and IPA (30 sec)
Process	Substrate Drying (use a single-wafer spinner)
Temperature	2500 rpm
Time	60 sec

Step 10: Electron-beam lithography, defining chip-scale EBL marker

Process	Spin coating PMMA
Spin Speed	6000 rpm
Time	45 sec
Thickness	170 nm
Process	Softbake
Temperature	160 $°C$
Time	3 min
Process	Electron-beam Exposure (Raith EBPG5150)
Dose	1100 μC cm^2
Current	100 nA
Process	PMMA Development using MIBK:IPA (1:3 vol %)
time	45-60 sec
Post Treatment	rinse with IPA for 30 sec
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec

Step 11: Sputtering and Lift-off 2

Process	Sputtering (T'COathy)
Material 1	Titanium
Thickness	5 nm
Materials 2	Platinum
Thickness	45 nm
Process	Lift-off (WB 11)
Chemical	Acetone
Time	10 min (or longer if the metal is not completely lifted-off)
Post Treatment	Rinse with Acetone (30 sec) and IPA (30 sec)
Process	Substrate Drying (use a single-wafer spinner)
Spin Speed	2500 rpm
Time	60 sec

Step 12: Dicing

Process	Dehydration Bake (on a hotplate)
Temperature	120 °C
Time	5 min
Process	Clean using DI wSpin Coating HMDS (WB 21)ater
Spin Speed	4000 rpm
Time	30 sec
Process	Spin Coating Olin OiR 907-17
Spin Speed	4000 rpm
Time	30 sec
Process	Wafer Dicing
Size	1x1 cm^2 (along the grid in Figure 3.2a)

3.5.2 Nano-scale Process Flow

Step 1: Grab and clean the chip

Process	Get 1x1 cm^2 chips from the diced wafer
Batch Tip	It is recommended to perform the nano-scale fabrication in batches of at least 4 chips.
Process	Resist removal (WB 11)
Chemical 1	Acetone
Time	5 min
Chemical 2	IPA
Time	30 sec
Process	Substrate Drying (use a nitrogen gun)
Time	Until Dry

Step 2: Electron-beam lithography, defining the electrodes

Process	Spin coating PMMA
Spin Speed	6000 rpm
Time	45 sec
Thickness	170 nm
Process	Softbake
Temperature	160 $°C$
Time	3 min
Process	Electron-beam Exposure (EBPG5150)
Dose	1100 μC cm^2
Current	10 nA
Dose Size Factor	above 200 nm use 1x, between 200 nm and 50 nm use 5x and below 50 nm use 10x
Process	PMMA Development using MIBK:IPA (1:3 vol %)
time	45-60 sec
Post Treatment	rinse with IPA for 30 sec
Process	Substrate Drying (use a nitrogen gun)
Time	Until Dry

Step 3: Electron-beam Evaporation + Lift-off

Process	Surface cleaning using a UV-Ozone reactor
Time	5 min
Process	Native oxide strip in 1% HF (WB 13)
Time	10 sec
Process	Cascade Rinse (Clean using DI water)
Process	Substrate Drying (use a nitrogen gun)
Time	Until Dry
Process	Electron-beam Evaporation (BAK600)
Material stack	1.5 nm Titanium/ 25 nm Palladium for Boron and 25 nm of aluminium for Arsenic
Process	Lift-off (on a Hotplate) (for Arsenic aim at performing RIE (step 4) a.s.a.p. after lift-off)
Chemical	Dimethyl sulfoxide (DMSO)
Temperature	90 °C
Time	15-30 min
Process	Ultrasonication (WB 11)
Temperature	Transfer from the hotplate at 90 °C to a sonication bath at 80 °C
Frequency	80 Hz
Power	30 W
Time	Batches of 1 min until lift-off is finished.
Post treatment	Rinse with Acetone (30 sec) and IPA (30 sec)
Process	Atomic Force Microscopy (AFM)
Needed information	The Electrode Height

Step 4: Reactive Ion Etching

Process	Surface cleaning using a UV-Ozone reactor (only perform the cleaning (ozone+HF) for Boron devices as it will remove the aluminium)
Time	5 min

Process	Native oxide strip in 1% HF (WB 13)
Time	10 sec
Process	Cascade Rinse (Clean using DI water)
process	Reactive Ion Etching (TEtske)
extra info	Each time use one of the chips in the batch to optimise the time below for the desired etching depth using the AFM (80 nm for Boron and 30 nm for Arsenic)
Pressure	10 mTorr
Power	25 W
Time	3-7 min for Boron and 1-3 min for Arsenic.
Chemical 1	CHF_3, 25 sccm
Chemical 2	O_2, 5 sccm
Other Chemicals	0 sccm
Process	Atomic Force Microsopy
Needed information	Etch depth, use the previously save electrode height.

References

[1] S. Fansilla. *Introduction to Microfabrication.* Wiley, 2 edition, 2010. ISBN 9780470749838.

[2] K. Abbas. *Handbook of Digital CMOS Technology, Circuits, and Systems.* Springer, 1 edition, 2020. ISBN 978-3-030-37194-4.

[3] M. Wang. *Lithography.* Intech, 1 edition, 2010. ISBN 9780470471555.

[4] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.C. Ruiz Euler, H. Broersma, P.r A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel. Classification with a disordered dopant-atom network in silicon. *Nature*, 577(7790):341–345, Jan 2020. doi: 10.1038/s41586-019-1901-0.

[5] Brigham Young University. Diffused ion implantation profile calculator and graph, 2022. URL `https://cleanroom.byu.edu/implantcal`. Last accessed 24 January 2022.

[6] R. C. Jaeger. *Introduction to Microelectronic Fabrication Volume V of Modular Series on Solid State Devices.* Prentice-Hall, 2 edition, 2002. ISBN 0201444941.

[7] Ion Beam Services, 2022. URL `http://www.ionbeamservicesuk.com/`. Last accessed 24 January 2022.

[8] B. El-Kareh. Silicon devices and process integration: Deep submicron and nanoscale technologies. Springer, 2009. ISBN 9780387367989.

[9] C. Zheng. *Nanofabrication, Prinsciples, Capabilities and Limits.* Springer, 2 edition, 2008. ISBN 978-1-4419-4536-5.

[10] J. Wildeboer. *Inverstigating the Intelligence Scaling of Neurmorphic Chips.* Master Thesis, 2020.

[11] R. Doering and Y. Nishi. *Handbook of Semiconductor Manufacturing Technology.* CRC press, 2 edition, 2008. ISBN 1574446754.

[12] Electricalvoice. Current to voltage converter - applications, 2022. URL `https://electricalvoice.com/current-to-voltage-converter-applications/`. Last accessed 6 March 2022.

[13] National Instruments. Ni-daqmx python api, 2022. URL `https://nidaqmx-python.readthedocs.io`. Last accessed 6 March 2022.

4

Classification with a Disordered Dopant-Atom Network in Silicon

Classification is an important task at which both biological and artificial neural networks excel[1, 2]. In machine learning, nonlinear projection into a high-dimensional feature space can make data linearly separable[3, 4], simplifying the classification of complex features. Such nonlinear projections are computationally expensive in conventional computers. A promising approach is to exploit physical materials systems that perform this nonlinear projection intrinsically, because of their high computational density[5], inherent parallelism and energy efficiency[6, 7]. However, existing approaches either rely on the systems' time dynamics, which requires sequential data processing and therefore hinders parallel computation[5, 6, 8], or employ large materials systems that are difficult to scale up[7]. Here we use a parallel, nanoscale approach inspired by filters in the brain[1] and artificial neural networks[2] to perform nonlinear classification and feature extraction. We exploit the nonlinearity of hopping conduction[9, 10, 11] through an electrically tunable network of boron dopant atoms in silicon, reconfiguring the network through artificial evolution to realize different computational functions. We first solve the canonical two-input binary classification problem, realizing all Boolean logic gates[12] up to room temperature, demonstrating nonlinear classification with the nanomaterial system. We then evolve our dopant network to realize feature filters[2] that can perform four-input binary classification on the Modified National Institute of Standards and Technology handwritten digit database. Implementation of our material-based filters substantially improves the classification accuracy over that of a linear classifier directly applied to the original data[13]. Our results establish a paradigm of silicon-based electronics for small-footprint and energy-efficient computation[14].

This Chapter is based on: T. Chen, J. van Gelder, B. van de Ven et al. Classification with a disordered dopant-atom network in silicon. Nature **577**, 341–345 (2020). doi: 10.1038/s41586-019-1901-0

Contributions: Measurements and data analysis.

4.1 Introduction

Doping is a crucial process in semiconductor electronics, where impurity atoms are introduced to modulate the charge carrier concentration. Conventional semiconductor devices operate in the band regime of charge transport, where the delocalization of the charge carriers gives rise to high mobility and a linear response to an applied electric field. At sufficiently low doping concentration and temperature[9, 15], however, delocalization is lost, and carriers move sequentially from dopant atom to dopant atom. This is referred to as the hopping regime[10, 11, 16], which exhibits higher resistivity and nonlinearity. Nonlinearity is often undesired, but it is a valuable asset for unconventional computing, that is, for systems that do not follow the Turing model of computation[6, 7, 8, 17, 18, 19]. Rather than excluding nonlinearity, we can exploit it[12] and manipulate our physical system with artificial evolution to solve computational problems[17]. This evolution in materio has been used, for example, for frequency distinguishing by liquid crystals[18] and robot control with carbon nanotubes[19]. We recently showed that a disordered network of gold nanoparticles acting as single-electron transistors can be evolved into any Boolean logic gate at sub-kelvin temperatures[12]. By exploiting the physics of materials for computation at the nanoscale through evolution, we may realize systems with unprecedented computational density and efficiency that are too complex to design[20].

Here, we fundamentally advance our previous work[12] by expanding the functionality, exploiting the well established platform of silicon technology and demonstrating operation up to room temperature. According to Cover's theorem[4], complex, linearly inseparable classification problems, when nonlinearly and sparsely mapped to a higher-dimensional space, can transform into linearly separable problems. The essence of this nonlinear mapping is illustrated in Figure 4.1a for the XOR classification problem. To save resources, this projection is often done implicitly by using kernel functions in machine learning, that is, without explicit computation of high-dimensional coordinates[3]. In artificial neural networks (ANNs), the nonlinear projection is learned by adjusting internal weights, traditionally through backpropagation, leading to powerful feature extractors[2]. However, emulating ANNs with conventional complementary metal–oxide–semiconductor (CMOS) technology is known to be power-inefficient[21], and CMOS scaling is not keeping pace with ANNs[14]. To avoid the area and power costs of emulating neurons and synapses, reconfigurable[2] material systems with intrinsic complexity and diversity of nonlinear operations[6, 22, 23] are strongly sought after.

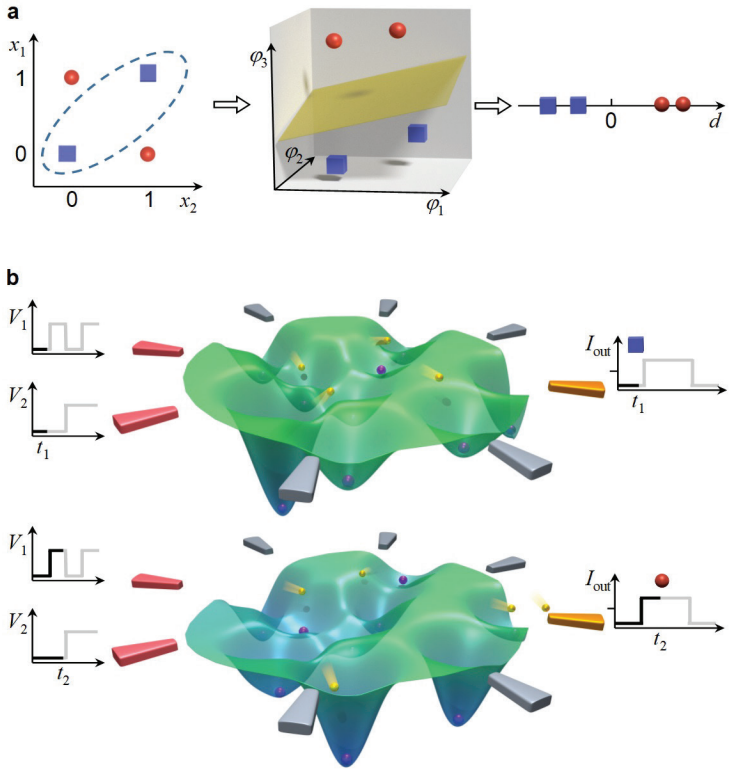


Figure 4.1: Simplifying classification by nonlinear projection. a, In the XOR classification problem two classes of data (red circles for (1,0), (0,1) and blue squares for (0,0), (1,1)) cannot be linearly separated in two dimensions (x_1, x_2; left). When nonlinearly transformed to three dimensions (ϕ_1, ϕ_2, ϕ_3; middle), the data can be linearly separated according to their distances d (right) to a decision boundary (yellow plane in the middle panel). b, Schematic representation of the potential landscape of the dopant network. In the hopping regime, the potentials of N dopants (purple spheres) span a highdimensional feature space. Yellow spheres represent charge carriers. The voltage–time (V–t) diagrams on the left schematically show the voltage combinations applied to the input electrodes (red), affecting the potential landscape and projecting information nonlinearly to the feature space. Note the difference between the potential landscapes in the top and bottom panels for different input voltages. The characteristics of the output current (yellow electrode) are tunable by the control voltages (grey electrodes).

4.2 Dopant Networks

Our system consists of a disordered network of boron dopants in silicon (Si:B) and is illustrated in Figure 4.2a, b. The boron atoms were implanted in n-type silicon with a concentration of $2 \cdot 10^{19}\ cm^{-3}$ at the surface (section 4.6.1, Figure 4.5). A 300-nm-diameter active region was defined by eight electrodes. The central silicon region was etched (about 80 nm deep) so that the boron concentration at the receded surface was reduced to about $5 \cdot 10^{17}\ cm^{-3}$, as confirmed by secondary-ion mass spectroscopy. The current–voltage (I–V) characteristics (Figure 4.2c, Figure 4.6) become increasingly nonlinear with decreasing T, and can be modelled as electric-field-activated hopping conduction at low temperatures (section 4.7.1, 4.7.2). The network's potential landscape (Figure 4.1b) depends in a highly nonlinear way on the input and control voltages, and spans a high-dimensional space. The output current is determined by this complex potential landscape. The nonlinear projection is realized when a combination of two or more input voltages is converted to an output current.

To identify the charge transport regimes, we focus on the low-bias conductance $G = dI_D/dV_{SD}|_{V_{SD}=-10mV}$[11], where I_D is the drain current and V_{SD} is the source–drain voltage:

$$G(T) = G_b e^{(-\epsilon_b/k_B T)} + G_h e^{-(T_h/T)^p} \tag{4.1}$$

where the first term describes band (b) conduction and the second term describes hopping (h) conduction. G_b and G_h are pre-factors with a much weaker temperature (T) dependence than the exponential terms, ϵ_b is the dopant ionization energy, T_h is a characteristic temperature of hopping conduction and k_B is the Boltzmann constant. The exponent p depends on the specific hopping model[11]. The resistance $R = 1/G$ as a function of inverse temperature $1/T$ is shown in Figure 4.2d. At $T > 250$ K, hole-band conduction dominates. The extracted ϵ_b is about 130 meV, three times larger than the value of boron in bulk silicon, about 45 meV. We attribute this increased ionization energy to dopant deactivation[24, 25]: for hydrogen-like dopants near the silicon surface, the decreased dielectric screening leads to stronger electron confinement, and therefore a larger ionization energy. We adopt the method proposed

by Zabrodskii et al.[15] to distinguish the hopping regime and extract p (section 4.6.2). For 70–160 K, we find $p = 0.342 \pm 0.023$, in agreement with $p = 1/3$ predicted for two-dimensional Mott variable-range hopping (Mott-VRH)[11, 26] (Figure 4.2e). The two-dimensional nature implies that the dopants participating in transport are located close to the silicon surface, because the hopping resistance increases exponentially with inter-dopant distance[11], which is lowest near the surface. This is consistent with the dopant deactivation observed in the band-conduction regime. Above about 160 K, band conduction starts to contribute, becoming dominant above about 250 K.

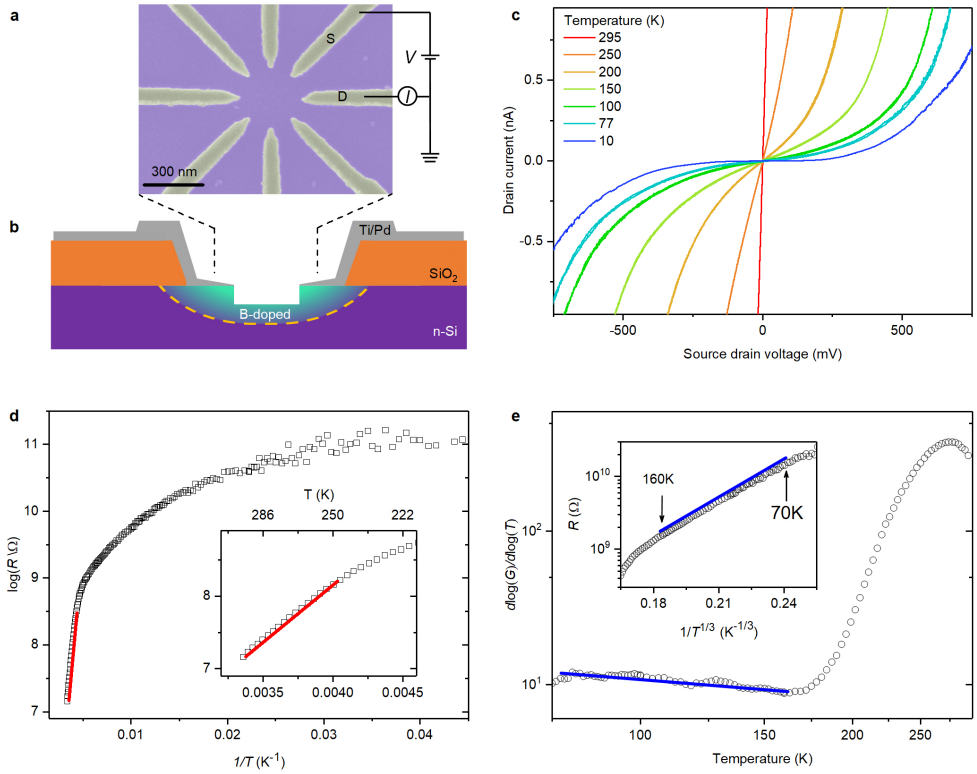


Figure 4.2: Device structure and charge transport mechanism. a, Scanning electron microscope image, indicating the source (S) and drain (D) contacts for I–V measurements. b, Schematic crosssection, illustrating the doping profile and the p–n junction (yellow dashed line). c, I–V characteristics at different temperatures (T) showing nonlinear behaviour below about 250 K. d, Resistance R versus inverse temperature at $V_{SD} = 10\ mV$. Band transport is observed for 250–295 K (indicated by the red line in the main figure and the inset, which shows the high-T region). e, Logarithmic derivative of the low-bias conduction G with respect to T. The linear segment for 70–160 K indicates hopping conduction (blue line). Inset, semi-logarithmic plot of R versus $1/T^{1/3}$, indicating two-dimensional variable-range hopping for 70–160 K (blue line) with Th $= 7.7 \cdot 10^4$ K, falling well within the range reported for Mott's VRH model[16].

4.3 Re-configurable Boolean Logic

To demonstrate classification in the hopping regime (Section 4.7.3–4.7.7), we followed the evolutionary approach of ref. [12] (section 4.6.6) and configured the system into Boolean logic (Figure 4.3a–c, Figs. 4.8–4.7) at 77 K. The working-temperature win-

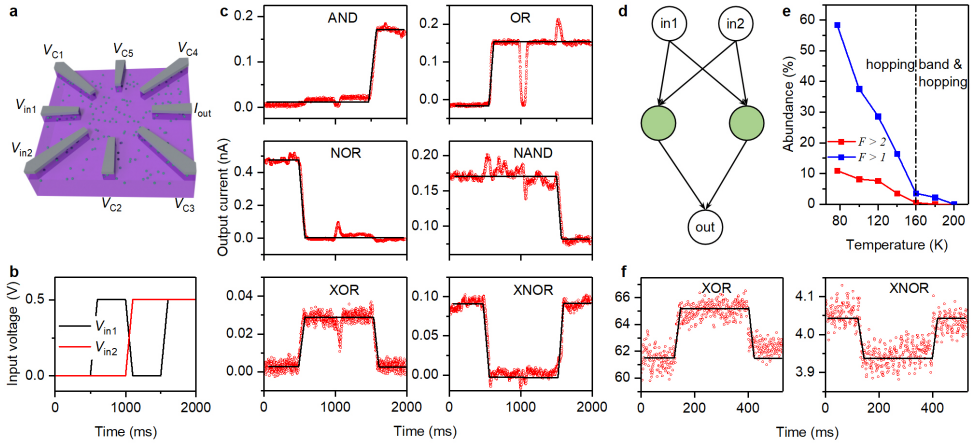


Figure 4.3: Evolution of Boolean logic. a, Schematic electrode configuration, indicating input voltages (V_{in1}, V_{in2}), control voltages (V_{c1}–V_{c5}) and output current (I_{out}). b, Input waveforms. The logic 0 and 1 are represented here by two different voltages, 0 V and 0.5 V, respectively (see also section 4.7.6). c, Major Boolean logic gates at 77 K (experimental current values in red, desired output normalized to the experimental data in black). We reproduced all Boolean logic gates in seven devices. d, ANN with two hidden neurons (green filled circles) emulated by the dopant network device. The ANN requires six (linear) weight multiplications, three (linear) summations and three (nonlinear) activations. e, Total abundance of logic gates (defined in section 4.6.5) as a function of temperature. The dashed line marks the onset of band conduction. The blue and red curves correspond to fitness thresholds of $F > 1$ (noise level) and $F > 2$, respectively. f, XOR and XNOR gates evolved at room temperature with a backgate voltage of about 12 V.

dow for a set of control voltages (about 30 K) is approximately 15 times wider than in our previous nanoparticle system[12] (about 2 K). The retention period of the gates is over two months in liquid nitrogen, and the device characteristics remain virtually unchanged after thermal cycling, indicating the robustness of the dopant network. Boolean logic represents a prototypical two-input binary classification problem[3], and the XOR classification problem is a poignant example of a single-layer perceptron's inability to solve problems with linearly inseparable vectors[27]. Hence, solving the linearly inseparable X(N)OR problem demonstrates the system's separation ability[3, 22, 23] (Figure 4.1a).

As realizing all Boolean logic gates with a standard ANN requires at least one hidden layer of two neurons[3] (corresponding to nine linear and three nonlinear operations), our dopant network can be considered to emulate at least such a neural network in hardware (Figure 4.3d). Importantly, the dopant network has only a 300-nm-diameter

footprint and an average power consumption of about 1 μW (section 4.6.7). Using established monolithically integrated readout circuits (section 4.6.4, Figure 4.11), the bandwidth of the readout circuitry can be increased from 40 Hz in our current setup to over 100 MHz. With optimization (section 4.6.4 and section 4.7.8), the energy efficiency of the dopant network at 77 K is projected to exceed 100 tera-operations per second per watt ($TOPs^{-1}W^{-1}$), where OP is one typical linear operation of a neural network[28]), one order of magnitude higher than a state-of-the-art customized CMOS neural network accelerator[29] (Section 4.7.8, 4.7.9, Figure 4.12b). However, recent simulation results indicate that bandwidth might be limited by dopant network [30]. Kinetic Monte Carlo simulations of the hopping process show that the bandwidth of the dopant network is 1 MHz. The projected energy efficiencies for both bandwidths are presented in section 4.6.4 and .

To investigate the correlation between the functionality of our devices and the transport mechanism, we performed random searches with 10,000 sets of control voltages as a function of temperature. We define the total abundance A, representing the overall probability of finding Boolean logic, with two fitness F thresholds for each logic gate[12] (section 4.6.5). For both fitness thresholds $F > 1, 2$, the total abundance drops to below 5% when band conduction sets in at around 160 K (Figure 4.3e). Hence, functionality is highly correlated to the hopping regime.

Led by this correlation, we tried to increase the operating temperature by suppressing band conduction. With increasing temperature, dopants near the p–n junction (Figure 4.2b) are expected to be ionized first, as they are less subject to deactivation than dopants far away from the junction. By depleting the junction using a back-gate, we indeed observe nonlinearity, and can evolve all six major logic gates at room temperature (Figure 4.3f, Figure 4.10). The confirmed correlation between functionality and the charge transport mechanism can serve as a guiding tool towards robust functionality at room temperature.

4.4 Handwritten Digit Classification

To demonstrate the ability of our device to perform more complicated classification tasks, we performed four-input binary classification in the form of filtering 16 2×2 black (1) and white (0) pixel features, as shown in the inset of Figure 4.4a. The four pixel values are encoded as four input voltages to our dopant network, together with three control voltages and one output current. We use the three control voltages to evolve a single network into 16 different filters at 77 K. Each filter should make one of the 16 features distinguishable from all the others, which is realized by evolving the dopant network such that it yields the maximal or minimal output current for that specific feature (Figure 4.4a, Figure 4.13). If we feed a feature to a group of 16 filters, each of which distinguishes one feature, then the 4-dimensional data are mapped to a 16-dimensional vector, and each feature vector is separated from the others in one of the dimensions (section 4.7.9).

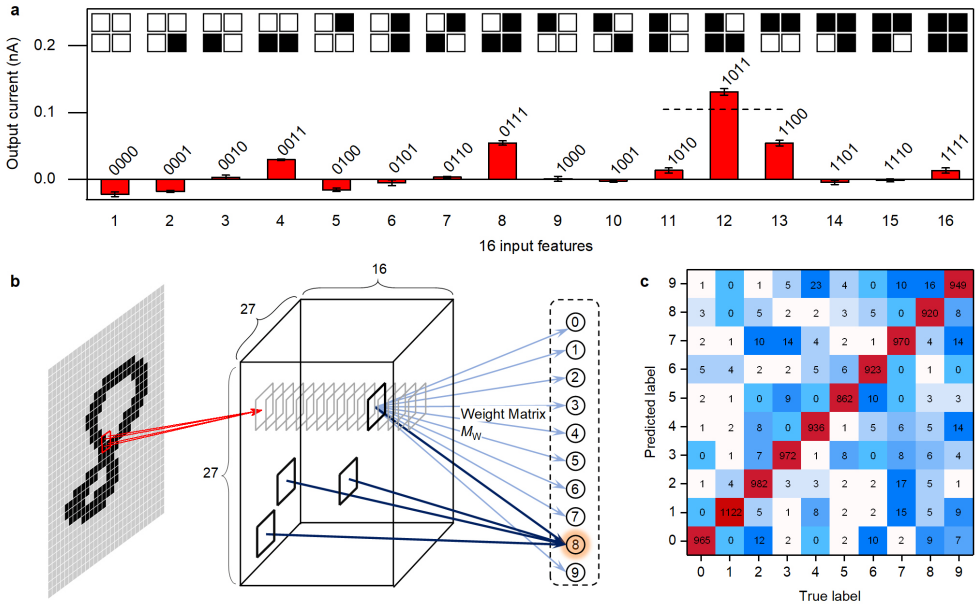


Figure 4.4: Feature filtering and handwritten digit classification. a, Current response of one of the 16 filters. The 2×2 pixel black/white patterns (inset) are represented by '0000', '0001', ..., '1111', with black (1) and white (0) mapped to input voltages 0.5 V and 0.5 V, respectively. The output current of this filter is maximal when the '1011' pattern is presented. Error bars represent the standard deviation of ten tests. b, Feature mapping for digit recognition. Specific filters are activated (bold dark squares) depending on the features presented to them.For clarity, most of the $27 \times 27 \times 16$ filters are not shown. The output of the filters is obtained from the experimental data shown in a and Figure 4.13. The ten output nodes, representing digits 0 to 9, are connected to the filters through a weight matrix M_W of a linear classifier. c, Confusion matrix of classification with the 10,000 MNIST test dataset, showing that 96.0% of the digits are correctly classified.

Our approach allows the separation of data by evolving filters that are capable of processing data in parallel and with high throughput. Compared with optical networks, which also allow parallel processing, our dopant networks feature tunability and have much smaller dimensions: about 100 nm instead of centimetres[7]. Taking advantage of the separation ability of our nanomaterial system, we used the evolved filters as the core ingredient to classify the Modified National Institute of Standards and Technology (MNIST) digits[13]. The whole classification procedure consists of a feature mapping layer of the evolved filters inspired by the convolutional neural network[2], followed by a linear classifier in a traditional computer, which can in principle also be realized in materio[31] (Figure 4.4b). The 28×28 greyscale pixels of

each MNIST digit are converted to black and white using a threshold and divided into 2 × 2 pixel receptive fields (overlapping in one row/column with neighbours). The receptive fields feed their signal to the cluster of 16 filters, each filtering out one of the features. The (28 × 28)-dimensional MNIST data are hence mapped onto (27 × 27 × 16)-dimensional feature vectors. The linear classifier then converts these high-dimensional feature vectors to a 10-dimensional output by a weight matrix MW (Figure 4.4b), obtained by pseudo-inverse learning[32] with the 60,000 MNIST training data (section 4.6.8). The largest of the ten outputs finally determines the recognized digit.

Application to 10,000 test digits shows 96.0% accuracy (Figure 4.4c, section 4.7.9, Figure 4.13), which is better than the accuracy obtained with state-of-the-art physical reservoir computing[8] and optical networks[7]. We note that differences in the output current scales of the different filters are irrelevant, because the weight matrix will automatically compensate for those (section 4.7.9). We also simulated feature filters with ideal characteristics, which are only activated when presented with its corresponding feature (output 1 for target feature and 0 otherwise). The classification of the MNIST dataset with these ideal filters results in an accuracy of 96.2%. Therefore, as long as the data mapped to the feature space are sufficiently separated, a linear classifier can learn the decision boundaries. The underlying reason is that every complete set of independent vectors, be it orthogonal (ideal) or not, can represent other vectors by linear combination. This shows the power of our dopant network in making data linearly separable, owing to its intrinsic nonlinear transformation. The ability to separate data, when combined with an adaptable linear readout in a scaled-up system, can achieve universal computational power[8, 22, 23]. For instance, in ANNs, perceptrons can be cascaded to solve more complex problems[3]. This analogy strongly suggests that a system of interconnected dopant networks can address a much wider range of tasks, particularly because the computational power of a single dopant network is larger than that of a single perceptron (it can solve XNOR whereas a single perceptron cannot).

4.5 Conclusion

At the system level, we anticipate a number of necessary developments. First, the total evolution time of the filters, which scales linearly with their number, can be reduced (by a factor 106; see sections 4.6.8 and 4.7.7, 4.7.8). Besides competitive evolutionary approaches[33], we will also explore gradient-based methods[34]. Second, it will be highly advantageous to store the evolved control voltages locally, employing, for example, memristors[31] (section 4.7.8). Third, memristive technology is also suitable for in materio implementation of the linear classification step in our scheme with energy efficiency comparable to our material-based nonlinear feature filters. Fourthly, processing analogue instead of binary signals would be more natural for our devices. To filter more complex, non-binary features, such as edge detection by the brain[1],

more electrodes per device are needed and/or multiple devices need to be interconnected, so that more input signals can be processed in parallel. However, increasing the number of input signals also makes it more difficult to distinguish them from one another. This increases the difficulty of the task resulting in the need for more accurate tuning of the potential landscape of the dopant network. Fortunately, increasing the number of electrodes and or interconnecting multiple devices also allows for more control voltages per filter (at present, three) to improve the signal-to-noise ratio and is capable of more accurate tuning of the device. We should keep in mind that an increased number of control electrodes might also increase the required training time. Besides this, interconnecting multiple DNPUs will result in the need for gain, fan-out and IV-conversion. This will take up space and energy reducing the efficiency of the DNPU network. Lastly, for practical applications, room-temperature operation with long retention, low-voltage supplies and without a backgate is desired, which we deem possible by engineering the deactivation effect in a silicon-on-insulator-based system.

4.6 Appendix A: Methods

4.6.1 Samples

300 nm of thermal oxide was grown on an n-type silicon substrate (Figure 4.5a), in which 26×60 μm^2 implantation windows were defined by photolithography and wet etching. Another 35 nm of oxide was thermally grown in the implantation window to serve as a stopping layer (Figure 4.5b). After boron implantation (9 keV equivalent, $3.5{\cdot}10^{14} cm^{-2}$), and activation via rapid thermal annealing ($1,050°C$, 7 s; Figure 4.5c), the 35-nm stopping layer was removed by wet etching. The boron concentration near the silicon surface exceeds $2 \cdot 10^{19} cm^{-3}$ to ensure Ohmic contact with the electrodes, and decreases monotonically with depth (Figure 4.5h). After lift-off of the wire-bonding pads (1.5 nm Ti/40 nm Pd) defined by photolithography (Figure 4.5d), eight 1.5 nm Ti/40 nm Pd nanoelectrodes were patterned on top of the silicon by electron-beam lithography (Figure 4.5e). The devices were annealed at $160°C$ for 10 min to promote the metal/silicon contact quality. The silicon surface was further etched by reactive ion etching to reduce the boron concentration in the active gap area (Figure 4.5f, g; see also section 4.7.6). The surface was finally treated with mild oxygen plasma, followed by 1% HF etching to remove possible contaminants.

4.6.2 Charge Transport

Following Zabrodskii et al.[15], we introduce the logarithmic derivative $w = d(log(G))/d(log(T))$. From equation. (1), we see that if the hopping term $G_h e^{-(T_h/T)^p}$ dominates, $log(w) \approx log(p) + p(log(T_h) - log(T))$, and p can be derived from the slope of the $log(w)$–$log(T)$ curve (Figure 4.2e), thus allowing us to identify the exact hopping conduction model. For T < 70 K, the measurement noise level prevents unambiguous identification of the charge transport mechanism (Figure 4.2d), but probably VRH continues[35]. The charge transport behaviour described in the main text has been observed in the two devices we characterized.

4.6.3 Measurements

We conducted the charge transport measurements and evolution of logic gates at different temperatures in a customized flow cryostat. The cryostat is equipped with 12 coaxial cables to reduce capacitive cross-talk. We use a battery-powered electronics rack (IVVI rack and matrix rack; `http://qtwork.tudelft.nl`) composed of digital-to-analogue converters (DACs) and I/V converters for low-noise measurements (Figure 4.11). The output range of the DACs is from -2 V to 2 V. The I/V converter has four amplification settings, 1 $G\Omega$, 100 $M\Omega$, 10 $M\Omega$ and 1 $M\Omega$, each corresponding to a different measurement range. For measurements at cryogenic temperatures, 1 $G\Omega$ amplification is chosen as default, by which currents from -3.4 nA to 3.4 nA can be measured. The output of the I/V converter is sampledby a multimeter (Keithley

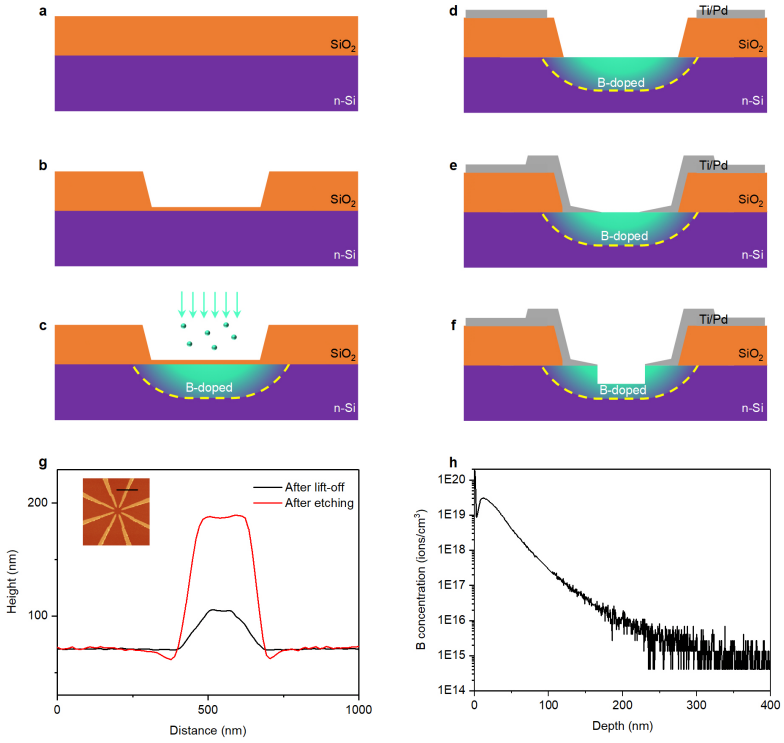


Figure 4.5: Fabrication steps and dopant concentration. a, Thermal oxidation. b, Implantation window definition and growth of 35 nm oxide. c, Ion implantation. d, Photolithography and contact pads lift-off. e, Electron-beam lithography and nano-electrodes lift-off. f, Reactive ion etching (RIE) of silicon. g, Height profile of the metal electrodes with respect to silicon before (black) and after (red) RIE etching. The etch depth of silicon is estimated by measuring the height change of the metal electrodes with respect to the silicon surface (indicated by the black line on the atomic force microscopy image in the inset, not to scale). Assuming that the metal is not etched by RIE, the etch depth of silicon is around 83 nm. h, Secondary ion mass spectroscopy of the boron dopant depth profile after implantation. On the basis of the etch depth, the boron concentration near the recessed silicon surface is of the order of $5 \cdot 10^{17} cm^{-3}$.

2000) or digitizer (ADwin-Gold II). For room-temperature evolution, the I/V converter amplification is set to 10 $M\Omega$, resulting in a current measurement range from -340 nA to 340 nA. The measurements are automated with either LabVIEW or

Python scripts. For fixed-temperature measurements, the devices were inserted into a liquid-helium (4.2 K) or liquid-nitrogen (77 K) dewar with a customized dipstick.

4.6.4 Readout Speed

In our system, the relaxation time of hopping conduction is less than 10 ns at 77 K and even smaller at higher temperatures (section 4.7.8), so it is not the dominant timescale in our present devices. Like in all measurements on resistive devices[36], the readout speed of our dopant networks is constrained by a large capacitive load (Figure 4.11b). The long, twisted pairs (about 3 m) as well as the filters of the matrix rack amount to a large load capacitance CL (about 4 nF) that limits the signal speed. With the existing setup, we have a bandwidth (cutoff frequency of the resistor–capacitor (RC) circuit in Figure 4.11b)

$$BW = \frac{1}{2\pi C_L(R_{out}||R_{Iv})} \approx 40Hz \tag{4.2}$$

where $R_{IV} = 1\ M\Omega$ is the input resistance of the I/V converter at 1 $G\Omega$ amplification, and the dopant network output resistance Rout is typically hundreds of MΩ (Figure 4.12d). By monolithically integrating a transistor-based readout circuitry close to the dopant network[36] (Figure 4.11c), we can reduce the capacitive load for fast readout, and also enable interconnection with other devices. With existing CMOS technology, the load capacitance can be easily reduced to below 1 fF, and the RC-related bandwidth can reach 160 MHz, or even more, by reducing RIV. Given a signal intensity (the difference between high and low output current levels; see 'Fitness functions' below), the signal-to-noise ratio (SNR) is predominantly set by the Johnson–Nyquist noise from $R_I V$, because its noise power is proportional to the bandwidth. Therefore, for a required SNR (computation precision), the bandwidth and the subsequent energy efficiency, are determined by the signal intensity (section 4.7.8). The signal intensity of our devices ranges from the order of 0.1 nA to the order of 1 nA (section 4.7.6), thus allowing over 100 MHz bandwidth (Figure 4.12a). However, recent kinetic Monte Carlo (KMC) simulations show that the bandwidth of the dopant network is likely to be 1 MHz[30]. The time of a single hop is indeed less than 10 ns at 77 K. However, relaxation of the network requires many hops. Each of these hops have a certain probability. For the KMC simulation they find that 10^6 KMC steps are required for the re-configurable Boolean logic to not be governed by noise. This equals approximately 1 μs or 1 MHz bandwidth. Therefore it is crucial to validate these simulations in hardware and increase this bandwidth by increasing the operation temperature or making smaller devices. For smaller devices, the reduced number of impurities could require less hops to stabilise.

4.6.5 Fitness Function

For Boolean logic gate evolution, the input sequences, representing the four input entries of truth tables (Figure 4.3b), were fed to the input electrodes (Figure 4.3a) after the control voltages were set. We monitored the output current waveform $\overline{Y}$ and fitted it with $\overline{Y} = m\overline{X} + C$, where $\overline{X}$ is the expected output waveform of a logic gate (logic high and low taking numerical values of 1 and 0, respectively). m is the proportionality factor and its value thus equals the separation of the high and low levels (signal intensity). C represents the offset. For each set of control voltages, a fitness is evaluated by $F = m/(\sqrt{r_{ss}} + kC)$, with r_{ss} being the fitting residual[12] and k an empirical constant. A larger k puts more emphasis on minimizing the offset C in the evolution process. For the evolution of logic gates, we found that there is minimal offset for $k = 0.2$.(Figure 4.3c).

In the random search of logic gates at different temperatures, k has been set to 0.01 to give the waveform shape more weight than the offset. Then, a fitness value of $F = 1$ implies that the signal intensity (related to m) almost equals the noise intensity (related to $\sqrt{r_{ss}}$), and a fitness value of $F = 2$ corresponds to more robust logic gates. Based on the fitness, we define the abundance of each gate. For the 10,000 output waveforms from a random search, we assessed the fitness of each output waveform for six major logic gates. In this way, each logic gate is associated with 10,000 fitness values. The abundance of a gate A_i (where i is AND, OR, NAND, NOR, XOR, XNOR) is defined as the number of fitness values larger than a threshold, divided by 10,000. The total abundance is then defined as $A = 1/(\sum_i (1/A_i))$. The fitness function for the feature filter evolution was defined as $F = |I_{out,i}|/(avg(|I_{out,j\neq i}|) + std(I_{out,j\neq i}))$, where $I_{out,i}$ is the output current corresponding to feature f_i, and avg and std stand for the average and standard deviation, respectively, of all the other feature outputs $I_{out,j\neq i}$. Here, i runs from 1 to 16.

4.6.6 Genetic Algorithm

The genetic algorithm mimics natural evolution. An initial generationof 20 genomes, with the length of each genome equal to the number ofcontrol electrodes, is first randomly generated and mapped to controlvoltages. The fitnesses of the 20 genomes are evaluated and ranked. Then the off-spring generation of 20 genomes is produced in the following way: (1) inheriting the five elite genomes (with highest fitnesses) from the previous generation; (2) cross-breeding of the elite genomes to produce five off-spring genomes; (3) mutation of the five elite genomes by a probability of 0.1, then cross-breeding with the five elite genomes to generate five other genomes; (4) cross-breeding of the five elite genomes with five random genomes to generate five other genomes. The genetic algorithm keeps iterating until it reaches a satisfactory fitness value (Figure 4.7a; see also section 4.7.7). A more detailed description of the evolution procedure is given in our previous work[12].

4.6.7 Power Consumption

To estimate the power consumption and energy efficiency of our device, we measured the static power consumption of the six major Boolean logic gates for four different input voltage combinations, so in total 24 configurations. To measure the current of the i^{th} (i running from 1 to 8) electrode, the voltage V_i (current I_i) is set (measured) by a source meter (Keithley 2401), while the voltages on the other electrodes are set by either the DACs (control voltages and input voltages) or an I/V converter (output electrode). For each of the 24 configurations, the total power P is calculated as $P = \sum_{i=1}^{8} V_i I_i$. The average power of the 24 configurations is found to be about 1 μW. Under operational conditions, the voltage changes on the electrodes are accompanied by charging and discharging of wire capacitances. As mentioned above ('Readout speed' section), the capacitances can be reduced to below 1 fF, making the dynamical power consumption negligible compared with the static power consumption. The static power consumption could be substantially reduced by using electrostatic electrodes (see also section 4.7.8).

4.6.8 Weight Matrix Training and Test

In the digit classification task, each 28×28 pixel digit is divided into 27×27 receptive fields of 2×2 pixels, overlapping by one row/column of pixels. The pixels of each receptive field are mapped to the 4 inputs of 16 filters (with their experimentally determined response), each of which filters 1 of the 16 distinctive 2×2 pixel features shown in the inset of Figure 4.4a. For the d^th digit in the Nd = 60,000 MNIST training database, we stack the $N_f = 27 \times 27 \times 16 = 11{,}664$ outputs of the filters in a feature vector $\boldsymbol{O}_d = (O_{d,1}, \ldots, O_{d,N_f})$. Combining the vectors $\boldsymbol{O}_d$ of 60,000 training digits together, we obtain an $N_d \times N_f$ output matrix $\boldsymbol{O} = (\boldsymbol{O}_1, \ldots, \boldsymbol{O}_{N_d})^T$. The true label of each digit is represented by a ten-dimensional label vector $\boldsymbol{L}_d$, whose elements are all zeros except for the $(l+1)^th$ entry being 1, where $l \in (0, \ldots, 9)$ is the true label of the d^th MNIST digit. Ideally, the weight matrix M_W converts the feature vector of a digit to its corresponding label vector $\boldsymbol{O}_d \boldsymbol{M}_W = \boldsymbol{L}_d$. So, in matrix form, $\boldsymbol{O}\boldsymbol{M}_W = \boldsymbol{L}$, where $\boldsymbol{L} = (\boldsymbol{L}_1, \ldots, \boldsymbol{L}_N)^T$. The weight matrix $\boldsymbol{M}_W$ has a dimension of $N_f \times 10$, and is simply obtained by $\boldsymbol{M}_W = O^+ \boldsymbol{L}$, where O^+ is the pseudoinverse of matrix O. Once the weight matrix is trained, we test it with the $N_t = 10{,}000$ MNIST test data. The feature vector of each test digit $\boldsymbol{O}_t, (t = 1, \ldots, N_t)$ is multiplied by the weight matrix to acquire the predicted label vector $\boldsymbol{P}_t, \boldsymbol{O}_t \boldsymbol{M}_W = \boldsymbol{P}_t$. The index of the maximal element of Pt minus one gives the predicted label. The accuracy is calculated as the ratio of the total counts of the correctly classified digits, that is, the sum of diagonal entries in Figure 4.4c, to the total number of test digits N_t.

4.7 Appendix B: supplementary Information

4.7.1 Surface States

Silicon atoms at the etched surface of the device are bonded with oxygen, forming a native oxide, which introduces localised surface states in the bandgap[37]. We rule out the direct involvement of these surface states in the functionality for the following reasons: Firstly, in the two devices in which the charge transport mechanism has been studied, the extracted ionisation energies are ~ 90 and ~ 130 meV. These are much smaller than the average energies of surface states, which are typically located at mid-gap, viz. a few hundred meV from the band edges[37, 38, 39]. Secondly, Figure 4.6a shows that at 4.2 K nonlinearity appears with incremental etching of the same device. If surface states would be the dominant factor, the IV characteristics would not depend on the etching time as each incremental etching step results in similar surface states, which contradicts the observation in Figure 4.6a. Moreover, 2D Mott variable-range hopping (VRH) has been widely observed in similar systems, such as in the inversion layer of metal-oxide-semiconductor field-effect transistors[26]. Therefore, we conclude that hopping conduction among dopants is the dominant transport mechanism.

4.7.2 IV Curves Fitting

In the main text, we elaborate on modelling charge transport at small electric field. Here we present more details of the electric-field-activated hopping by modelling the IV characteristics measured at different temperatures (Figure 4.2c). In various models of electric-field-activated hopping, the conductance G is found to scale as $G \propto e^{-(1/V)^{1/3}}$ in the strong-field regime[40]. Following Van Ancum et al.[40], we write the hopping conductance as $G(T,V) = G_h e^{-(k_B T_h/(k_B T + qEr))^{1/3}}$. Here, r is the average hopping distance, q the charge, and we assume a linear dependency of the electric field E on the applied voltage V. Taking into account the band conduction, we can then write the current as

$$I = VG_h e^{(-\frac{k_B T_h}{(k_B T + qEr)})^{1/3}} + VG_b e^{-\frac{\epsilon_b}{k_B T}} \tag{4.3}$$

In the high-field limit, and assuming $k_B T \ll Er$, we arrive at

$$I = VG_h e^{(-\frac{k_B T_h}{qEr})^{1/3}} + VG_b e^{-\frac{\epsilon_b}{k_B T}} \tag{4.4}$$

This model fits all the IV curves reasonably well, especially at large voltages, as shown in Figure 4.6d, thus suggesting that Eq. 1 in the main text provides an appropriate model for the conductance. In Figure 4.6e, we plot $log(G)$ as a function of $1/V^{(1/3)}$ at different temperatures to qualitatively reveal the effect of the temperature on the hopping conduction. The conductance values at temperatures below 140 K are bundled together in the high-voltage range (indicated by the black circle). Here the electric-field activation, i.e. the first term on the left-hand side of Eq. S2, dominates

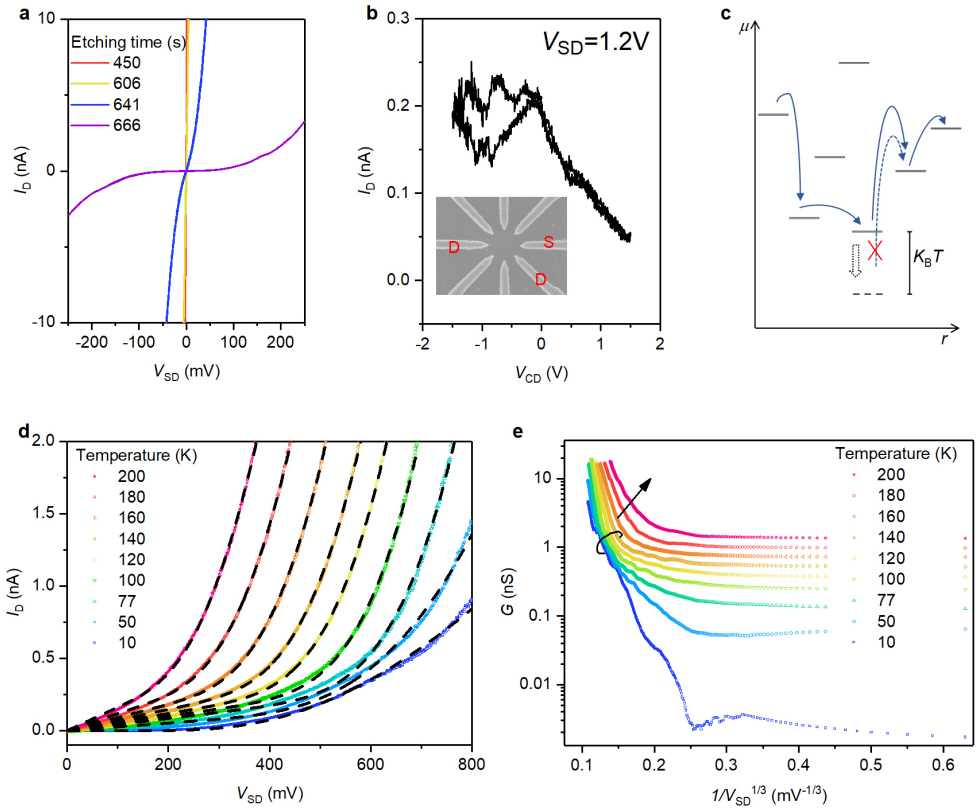


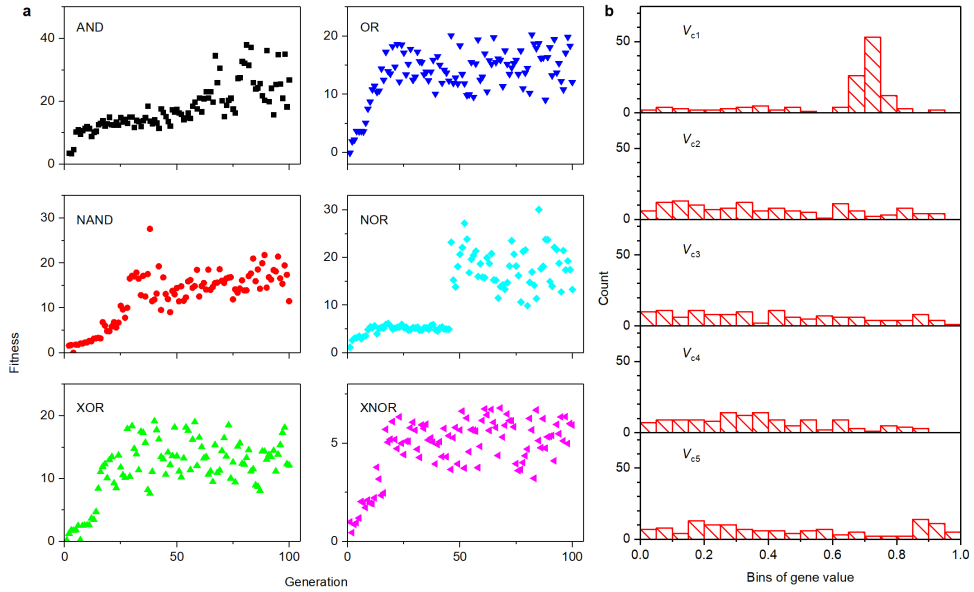
Figure 4.6: Nonlinear and tunable hopping conduction. a, I–V characteristics at 4.2 K with different total etching time by RIE. As the total etching time increases, the nonlinearity becomes increasingly prominent, signalling the dominance of hopping conduction. b, Drain current versus control voltage for constant source–drain voltage $V_{SD} = 1.2\ V$ at 4.2 K. The source (S), drain (D) and control (C) electrodes are shown in the inset. The hysteresis for negative gate voltage is probably due to charging of the other five floating electrodes. c, Schematic plot of electrochemical potential μ versus position r, illustrating the tunability. The solid lines represent impurity states and the arrows represent hopping of carriers among states. See section 4.7.3 for detailed discussion. d, Fitting the temperaturedependent I–V curves with the model described by equation (2) in section 4.7.2. Black dashed lines represent the fitted curves. e, Conductance versus the reciprocal of the cube root of the source–drain voltage at different temperatures. The black circle groups data at temperatures below 140 K. See also section 4.7.2 for more discussion.

the hopping current. For temperatures higher than 140 K, the curves in the high-field range increasingly depart from the bundle of low-temperature curves, which suggests that the influence of thermal energy k_BT (in Eq. S1) on conductance dominates over the electric field. Thus, increasing temperature decreases the tuneability of the hopping conduction by the electric field, and diminishes the nonlinear projection of data. This agrees with the observed correlation between charge transport and functionality discussed in the main text.

4.7.3 Hopping Conductance Tuning

The IV characteristics of the dopant network are highly tuneable by the control voltages. Figure 4.6b shows an exemplary three-terminal measurement at 4.2 K. Two neighbouring electrodes are used as source and drain electrodes, and another, oppositely located control electrode acts like a gate (inset of Figure 4.6b). In general, we note that the third control electrode may also draw current. Clearly, we observe negative transconductance (NTC) for positive control voltage, and a complex conductance behaviour for negative control voltage. We hypothesise that the network is energised by the source electrode (injecting electrons), and a percolation path is formed between the source and drain (Figure 4.6c). Along the percolation path, electrons hop consecutively along a series of dopant states. Increasing control voltage lowers the electrochemical potential of dopant states, which elevates or lowers certain energy barrier for hopping. The net effects over these changes in energy suppress or enhance the overall current, thus leading to NTC and complex conductance tuning.

To obtain a qualitative understanding of the influence of the control voltages on functionality, we consider data generated during a random search for the logic gates with 10,000 control voltage configurations. Figure 4.7b shows histograms of the values of five genes (control voltages)[12] yielding output waveforms with fitness $F > 1.5$ for the XNOR gate (see section 4.6.5). It can be seen that one control voltage (V_{C1}) peaks around a gene value of 0.7 (Figure 4.7b top panel), corresponding to $\sim 240mV$ (the control voltage range from $-600\ mV$ to $600\ mV$ is mapped onto the gene range 0-1). The other control voltages show no preferable ranges, indicating that there are multiple solutions for the XNOR gate, as long as the V_{C1} is located around the critical point. We note that it is the combination of control voltages that yields the solution, so although the distribution of $V_{C2} - V_{C5}$ is broad, they cannot be randomly chosen. This behaviour resembles what we have observed before in the gold nanoparticle networks[12]. A simple picture to understand this behaviour is that the device is mainly energised by one of the electrodes, most often a neighbouring electrode to the output. The other electrodes function as gates to tune the potential landscape, as explained above. Therefore, the functionality is most sensitive to voltages at the output electrode (grounded during measurements) and the energising electrode.



c

	V_{C1} (V)	V_{C2} (V)	V_{C3} (V)	V_{C4} (V)	V_{C5} (V)
AND	0.261	0.299	-0.619	-0.579	-0.221
OR	-0.266	-0.68	0.515	-0.175	-0.199
NAND	-0.298	0.177	0.515	0.244	-0.32
NOR	0.346	-0.25	0.328	1.18	-0.124
XOR	0.442	0.314	0.487	0.144	-0.209
XNOR	-0.279	0.018	0.038	-0.73	-0.256

d

	V_{C1} (V)	V_{C2} (V)	V_{C3} (V)
0000	0.60509	-0.34386	0.69212
0001	0.20454	-0.57179	0.64293
0010	-0.41105	0.29822	0.2447
0011	-0.82728	0.15518	-0.27901
0100	-1.29545	0.22605	1.08706
0101	0.54342	-0.86175	0.85753
0110	-0.32769	0.23459	0.26238
0111	-0.12659	0.24267	-0.49213
1000	-0.77783	-0.86176	1.07511
1001	-0.1088	-1.00302	1.03481
1010	-0.39624	0.06503	0.3365
1011	-0.07965	-0.30009	-0.43787
1100	0.02196	-0.21027	-0.82176
1101	-0.97294	0.21243	-0.92759
1110	0.01541	-0.07923	0.27283
1111	-0.38757	-0.54464	0.70877

e

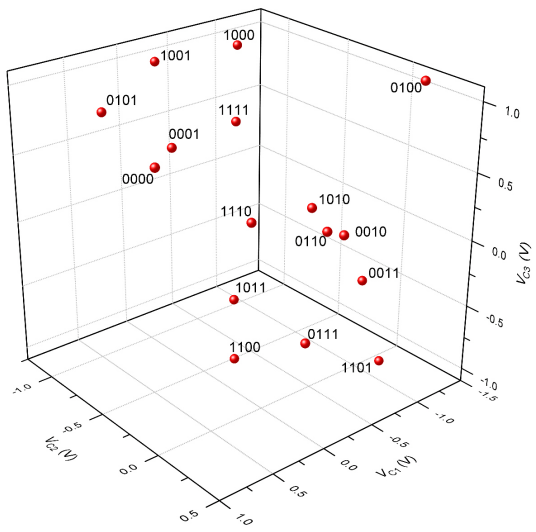


Figure 4.7: Convergence of genetic algorithm in the configuration space. a, Genetic algorithm convergence for the six major Boolean logic gates at 77 K. The best fitness of the 20 genomes is plotted against generation. b, Histograms of the control voltages that configure the dopant network to the XNOR gates with fitness F larger than 1.5. The first control voltage is prominently concentrated in a small range, but the others do not show a favourite range. The ranges of the five control voltages are (600, 600), (1,200, 1,200), (1,200, 1,200), (1,200, 1,200) and (600, 600). c, Control voltages for the six major logic gates. d, Control voltages for the 16 filters, which are visualized in e. The filters '0110' and '0010' have the smallest separation. See section 4.7.3 and 4.7.7 for more discussion.

To investigate the sensitivity of the functionality to a voltage change, we applied an offset voltage to the output electrode and found that the functionality is lost by an offset of $\pm 10\ mV$.

To see how the control voltages are distributed in the configuration space, we list the evolved control voltages for the six major logic gates and the feature filters in Figure 4.7c-d. For the filters, there are 16 sets of control voltages in a 3-dimensional configuration space, which are thus less separated than the 6 sets of control voltages in the 5-dimensional configuration space for the major Boolean logic gates. We visualise the filters' control voltages in a 3D plot in Figure 4.7e, where each dot represents the control voltages of one of the 16 filters. The most closely located control voltages are those for filters "0010" and "0110", with a separation vector of ($-84\ mV$, $64\ mV$, $-18\ mV$).

4.7.4 Thermal Stability

In order to study the thermal stability of the evolved logic gates, we re-tested a NAND gate evolved at 77 K at different temperatures. With the optimal control voltages for 77 K fixed, the temperature was swept down to 5 K, then up to room temperature, and cycled back to 77 K. The obtained fitness cycle is plotted in Figure 4.8b. Above 140 K, the output current clipped to compliance ($\pm 3.4\ nA$). At each temperature the fitness remains approximately the same during both the cooling and warming phase, demonstrating the robustness of the device. Figure 4.8b also shows that the optimal control voltages obtained at 77 K now remain applicable within a large range of about 65-95 K, where there is no significant drop in fitness (see section 4.6.5 for the definition of fitness). This is in contrast to our previous nanoparticle devices[12], in which the control voltages for one gate remain functional only within a temperature range of ≈ 2 K (Figure 4.13 in Ref. [12]). The larger temperature window achieved with our dopant network device should be sufficient for practical applications. We hypothesise that this range is correlated to the characteristic temperature Th for Mott-VRH, which can be further optimised[11].

4.7.5 Evolution of Logic Gates at 4.2 K, 140 K and Room Temperature

As discussed in the main text, the VRH is expected to continue to very low temperatures, and it should therefore be possible to evolve logic gates at low temperature. Figure 4.9a shows that these gates can indeed be evolved at a very low temperature of 4.2 K. Figure 4.9b shows the six major logic gates evolved at 140 K, which is close to the upper temperature limit of hopping conduction in our device. The evolved logic gates are becoming ill-defined in the sense that the 'low' and 'high' values in the output current are not well-aligned anymore, suggesting the tuneability is vanishing at higher temperature. The XOR and XNOR gates can be evolved only up to 160 K, where band conduction becomes important, which is consistent with the random search results in Figure 4.3e. In the case of evolution at room temperature

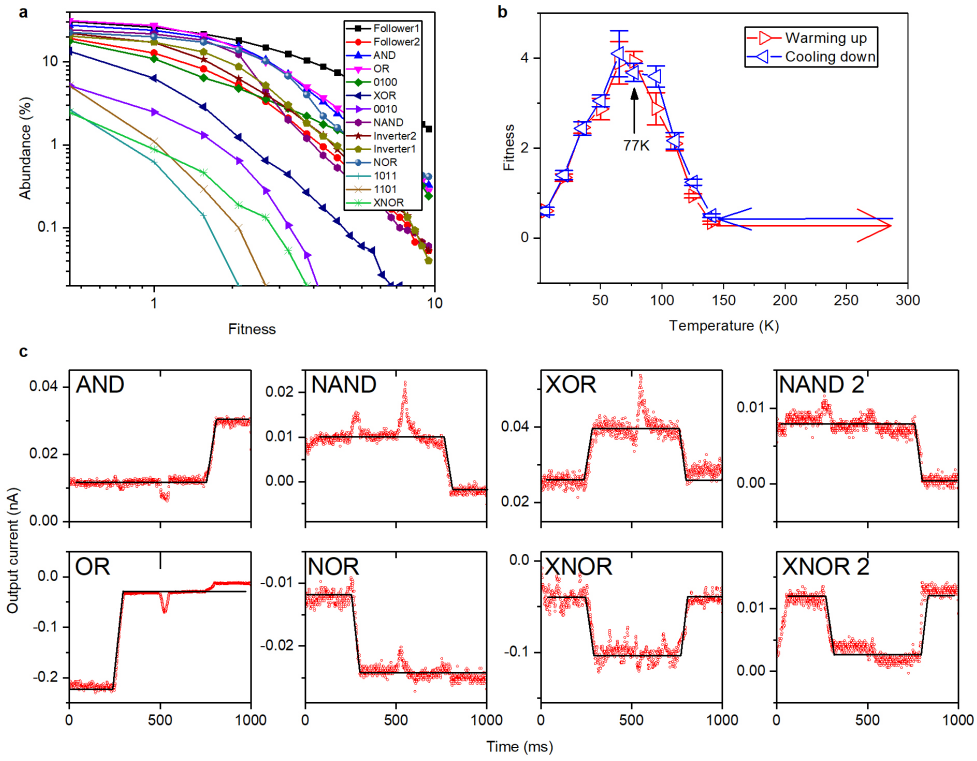


Figure 4.8: Evolved logic gates at 77 K. a, Abundance plot of 14 nontrivial truth tables at 77 K. From a search with 10,000 sets of randomly generated control voltages, we found all 16 possible truth tables that can be realized for a two-input–one-output configuration. b, Thermal stability of a NAND gate evolved at 77 K. Above 140 K, the output current clipped to compliance, and therefore the fitness was not extracted. The error bars represent the standard deviation of ten tests (see also section 4.7.4). c, Boolean logic gates evolved at 77 K in a device other than the one in Figure 4.3c. Red circles are experimental output currents, and black lines represent the normalized desired output currents. The left six panels show the six major logic gates evolved with input voltage levels 0 V and 0.5 V. The right two panels show a NAND and a XNOR gate evolved with input voltage levels of 0.25 V and 0.25 V, showing the adaptability of the dopant network to different voltage levels (see also section 4.7.6).

when depleting the p-n junction by applying a backgate voltage (Figure 4.10b), the tuneability is marginal and comparable to that at 140 K without a backgate voltage (Figure 4.9b). This indicates that further optimisation of the device fabrication is required for robust room-temperature operation.

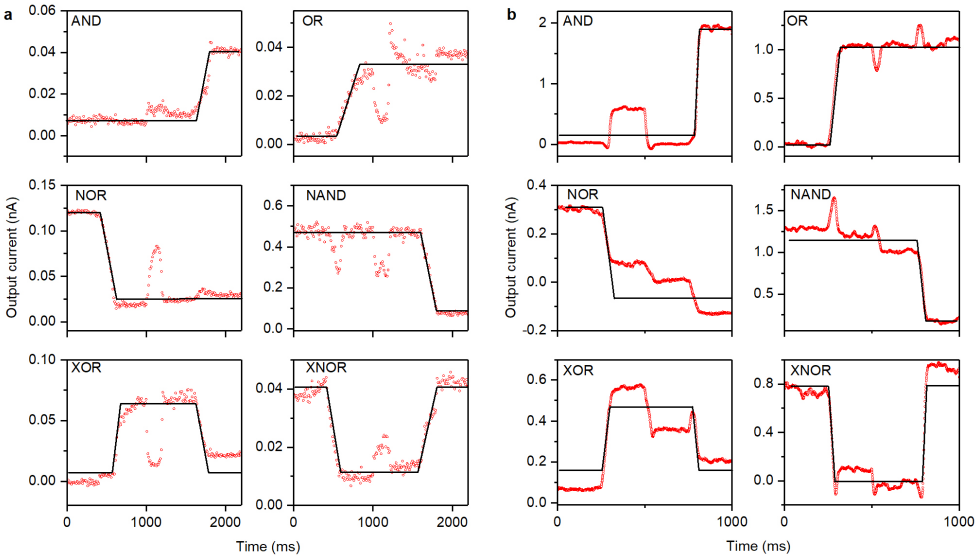


Figure 4.9: Evolution of logic gates at two ends of hopping conduction. a, Evolved logic gates at 4.2 K, at which the charge transport mechanism is still VRH (section 4.6.2). b, Evolved logic gates at 140 K. Red circles are experimental output currents and black lines represent the normalized desired output currents. See section 4.7.5 for a detailed discussion.

4.7.6 Reproducibility of Functionality

The characteristics of the seven devices that we investigated, and especially the output current scale (as well as signal intensity, the current separation between 'high' and 'low' in Boolean logic), vary from one device to the other (from 0.1 nA order to 1 nA order). Figure 4.8c shows the logic gates evolved at 77 K in a device different from the one in Figure 4.3c. As can be seen, the average separation between high and low current is approximately 3 times smaller than in Figure 4.3c. One of the factors contributing to this difference is the variation in dopant concentration. As discussed in the main text, we employed RIE etching to reduce the dopant concentration in the active region, which lacks precise control over the etching depth. The resulting variation can be minimised by optimising the fabrication procedure to remove the etching step.

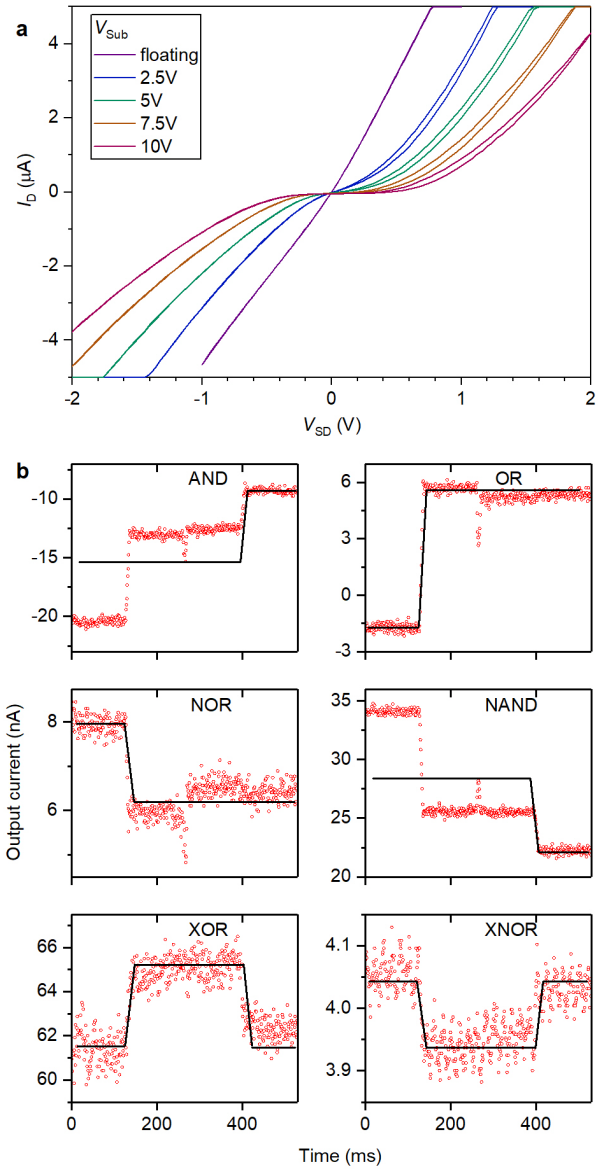


Figure 4.10: Backgate-induced nonlinearity and evolved logic gates at room temperature. a, A positive voltage V_{Sub} with respect to the drain voltage is applied to the n-type substrate (Figure 4.2b) to make the depletion region wider at the p–n junction, and to suppress the band conduction. b, Evolved gates at room temperature. Red circles are experimental outputs, and black lines represent the normalized desired outputs. The output current levels, and also the separation between these levels, are more than one order of magnitude larger than those of the logic gates evolved at 77 K, owing to the increased hopping conductance (section 4.7.3). The increased noise intensity is mainly due to the settings of the current measurement circuit (section 4.6.3).

We can evolve all devices to logic gates with the same control voltage range of $-2\ V$ to $2\ V$, the same output current range of $-3.4\ nA$ to $3.4\ nA$ (see Figure 4.3c, Figure 4.8c and Figure 4.9), and the same input voltage levels of $0\ V$ and $0.5\ V$. The devices are also able to adapt to different input values. For example, we can change the input voltage from $0\ V$ and $0.5\ V$ to $-0.25\ V$ and $0.25\ V$. The functionality can then be recovered by re-evolution, as demonstrated for the XNOR gate and NAND gate in Figure 4.8c (right two panels). In principle, the dopant network is more suitable for processing information in the analogue regime without need for binarization of the input dataset[41]. By interconnecting multiple devices or incorporating more electrodes on one device, we expect to solve real-life non-binary problems, such as filtering an edge.

4.7.7 Evolution Convergence and Speed

During the evolution process, the fitness of the logic gates increases until convergence, as shown in Figure 4.7a. Occasionally, an abrupt increase of fitness is observed (e.g. for the NOR gate), which implies that the control voltages suddenly change to reach a better fitness. After a sufficiently good control voltage set is found, the average fitness remains stable. Fitness fluctuations still occur due to stochastic variations in the output current. The output current fluctuations for the NAND, NOR, XOR and XNOR gates are also observed in Figure 4.3c. The current fluctuations are directly related to the nature of the hopping conduction, and can be reduced at higher dopant concentration and higher temperature[42].

For the seven investigated devices, it typically took 10-20 generations to evolve a logic gate with sufficient fitness, corresponding to ~ 15 min in our current measurement setup. The filters took a similar amount of generations to evolve. This translates to ~ 1 hour, which is a factor of four longer, since the number of possible input combinations is four times larger (16 instead of 4). Notably, the genetic algorithm has not been optimised and it can be expected to find solutions faster if its hyper-parameters are properly tuned to the task at hand.

The bottleneck in the evolution is the evaluation of a single control voltage configuration. As discussed in section 4.6.4, our measurement setup currently has a limiting bandwidth (cutoff frequency of the current readout circuitry) of about 40 Hz, corresponding to a time constant of $\sim 4ms$. We usually hold each input voltage 500 ms or longer to obtain sufficient data quality (see Figure 4.3b in the main text). However, the device can be reconfigured to logic gates much faster. We have successfully evolved functionality by setting the input holding time to 100 ms and the rise/fall time to 20 ms, suggesting that the reading time of our device can at least reach 30 times time constant ($30\times4 = 120\ ms$). We note that this reading time is still far from the theoretical limit. The output of a RC circuit can get to 99% of its steady-state value in 5 time constants[43]. The evolution speed will increase proportionally to the bandwidth, which offers plenty of room for optimisation. Meanwhile, the total

evolution time of filters increases linearly with their amount, making our approach scalable in terms of time consumption.

4.7.8 Bandwidth and Energy Efficiency Scaling

The energy efficiency is limited by the bandwidth (BW) of the readout circuitry (cut-off frequency of the RC low-pass filter in Figure 4.11b-c) for our dopant network, as this determines the throughput of operations. In turn, the bandwidth is limited by a) the relaxation time of hopping conduction, b) the Johnson-Nyquist noise from the I/V conversion resistor RIV (Figure 4.11c), $i_n^2/BW = 4k_BT/R_{IV}$ and c) the required signal-to-noise ratio (precision). As mentioned in section 4.6.4, the hopping relaxation time $\tau(T)$ is not the limiting timescale for the bandwidth in our present devices, because $\tau(T)^{-1} = \nu_{ph}e^{-(T_h/T)^{1/3}}$, where $\nu_{ph} \approx 10^{12}$ Hz, is the "attempt frequency"[44]. At $T = 77$ K, the relaxation rate is larger than 100 MHz. Furthermore, for a given signal intensity i_s (defined as the output current difference between high and low levels, related to m in section 4.6.5), and a required signal-to-noise ratio $SNR = i_s^2/i_n^2$, the maximal bandwidth is determined by

$$BW = \frac{i_s^2 \cdot R_{IV}}{4k_BT \cdot SNR} \tag{4.5}$$

We plot the bandwidth with respect to the signal intensity i_s at two required SNR values (black and red solid lines) in Figure 4.12a. Let us then take the average power consumption of the dopant network 1 μW (see section 4.6.7 and Figure 4.12d) to estimate energy efficiency. In analogy to a small ANN with 2 hidden units, capable of representing arbitrary Boolean logic, the operation of our dopant network is equivalent to performing 9 linear operations and 3 nonlinear operations when processing a single combination of input data. To be on the safe side, we assume that the complex operation of our network is equivalent to performing 10 linear operations, neglecting its intrinsic ability to perform nonlinear operations. This is a conservative estimate, as nonlinear operations[45] are generally more complicated to implement in hardware. The output reading time for our device is taken as 30 time constants, i.e. $30/(2\pi BW) \approx 5/BW$, also a realistic lower bound that has been experimentally confirmed (see section 4.7.7). In short, our device performs equivalently at least 10 linear operations in a time range at most $5/BW$, and therefore,

$$\text{Energy efficiency} \approx \frac{(10BW)}{(5 \cdot 10^{-6})} \approx \frac{2 \cdot 10^6 \cdot i_s^2 \cdot R_{IV}}{(4k_BT \cdot SNR)} \quad OP/s/W. \tag{4.6}$$

The energy efficiency, assuming that the bandwidth is limited by the readout circuitry, is plotted in Figure 4.12b. If we optimise the signal intensity to 5 nA (see section 4.7.6), then the energy efficiency is projected to reach 200 $TOP/s/W$ (TOP: tera operations) with $SNR = 10$. Further increase of signal intensity will not lead to higher energy efficiency, but higher SNR, unless the power consumption is reduced as discussed below.

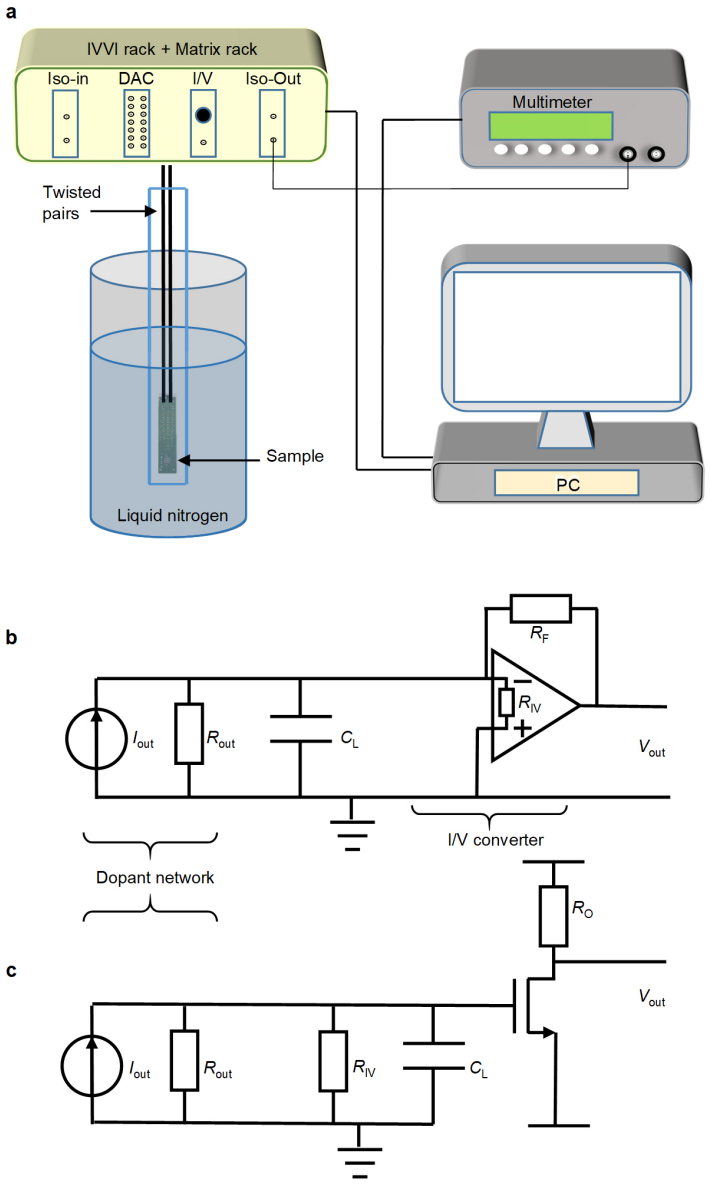


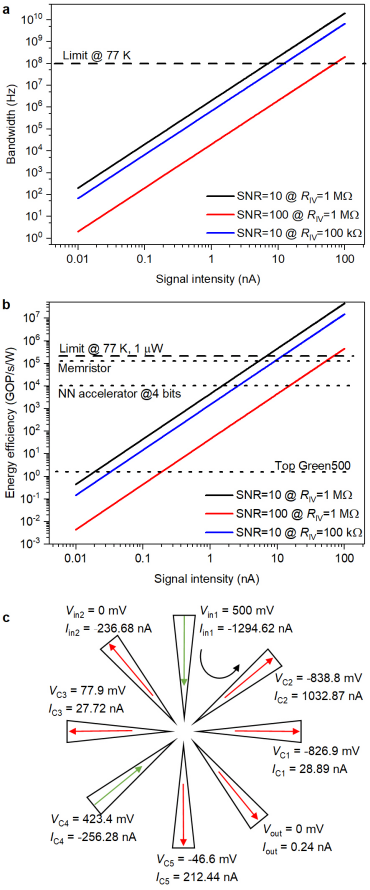
Figure 4.11: Measurement setup. a, Schematics of the existing measurement setup. b, Equivalent circuit of the current measurement setup. I_{out} and R_{out} represent the output current and output resistance of the device. C_L is the parasitic capacitance of about 4 nF. R_{IV} and R_F are the input resistance and feedback resistance of the I/V converter, respectively. c, Schematic of an integrated high-speed current reading circuit. Here, R_{IV} is a resistor to convert current to voltage, CL is the parasitic capacitance that can be reduced to below 1 fF. R_O is a resistor that sets the amplification.

As mentioned in section 4.6.4, recent KMC simulations indicating that the bandwidth of the dopant networks is likely to be 1 MHz[30]. The calculations above are still correct, we can now substitute the 100 MHz bandwidth from the readout circuitry with the 1 MHz found during the KMC simulation. For the signal intensity of 5 nA and SNR of 10 this results in a projected energy efficiency of 2 $TOP/s/W$.

We compare this result with the energy efficiency in other leading computation technologies. At the moment, the most efficient high-performance computer from the Green500 list is running at about 17 $GOP/s/W$[46], with high precision for floating-point operations. The application-specific integrated circuit (ASIC) CMOS-based neural network accelerator, which trades precision for energy efficiency, can reach 10 $TOP/s/W$ at 4 bits[29] ($SNR \approx 16$). As mentioned in the main text, the linear classifier can be implemented with memristors[47] in fully materials-based systems, so we also include the projected energy efficiency of memristors. Memristors are projected to reach over 100 TOP/s/W in weight multiplications taking into account the readout circuitry[47, 48].

The power consumption measurements demonstrate that there is a lot of room for improvement (see section 4.6.7 and Figure 4.12d). We found that very often a large current flows into one electrode and out from a neighbouring electrode (Figure 4.12c), because the current path between two adjacent electrodes is the shortest. This observation has two implications. Firstly, the influence of a voltage applied at one control electrode on the potential landscape is constrained by the voltages applied at neighbouring electrodes. Secondly, most power is consumed by these parasitic current flows. A method to solve this second problem is to couple the control electrodes electrostatically to the network, avoiding parasitic current flow, while remaining effective in tuning the potential landscape. In our previous work[12], it has been shown that electrostatic electrodes (such as a backgate) are also effective in evolving functionality.

We further estimate the power consumption of the peripheral circuitry required to provide the control voltages and to read the output signal. In principle, two resistive elements, made of memristors or floating gate transistors, can form a voltage divider and be programmed to provide a desired control voltage. With electrostatically coupled electrodes, parasitic currents are avoided and the potential divider consumes little power to supply the control voltage. In the read-out circuitry shown in Figure 4.11b, the quiescent current of the transistor can be as small as tens of nA, thus adding only tens of nano-Watt to the overall power budget when assuming a $\sim 1\ V$ supply voltage. Therefore, the overall power consumption can be kept to the same level as that of the dopant network itself.



d

NAND00	V (mV)	I (nA)	I_{std} (nA)	R_{diff} (MΩ)
Input 1	0.00	-81.81	0.84	1.22
Input 2	0.00	14.74	0.08	8.49
Control 1	-826.90	10.79	0.12	21.75
Control 2	-838.80	119.42	1.16	1.99
Control 3	77.90	24.33	0.22	7.99
Control 4	423.40	-339.16	1.50	0.43
Control 5	-46.60	251.61	1.47	0.96
Output	0.00	0.28	0.01	48.19
Power		0.26 μW		

NAND01	V (mV)	I (nA)	I_{std} (nA)	R_{diff} (MΩ)
Input 1	0.00	77.73	0.88	0.98
Input 2	500.00	-276.23	2.11	0.42
Control 1	-826.90	9.76	0.13	28.85
Control 2	-838.80	147.69	1.21	1.91
Control 3	77.90	48.95	0.22	2.74
Control 4	423.40	-203.48	1.12	0.57
Control 5	-46.60	182.3	1.06	1.38
Output	0.00	0.16	0.02	138.59
Power		0.36 μW		

NAND10	V (mV)	I (nA)	I_{std} (nA)	R_{diff} (MΩ)
Input 1	500.00	-1294.62	4.09	0.23
Input 2	0.00	236.68	2.06	0.75
Control 1	-826.90	28.89	0.25	12.94
Control 2	-838.80	1032.87	5.09	0.63
Control 3	77.90	27.72	0.21	6.37
Control 4	423.40	-256.28	1.96	0.52
Control 5	-46.60	212.44	1.39	1.11
Output	0.00	0.24	0.02	105.88
Power		1.65 μW		

NAND11	V (mV)	I (nA)	I_{std} (nA)	R_{diff} (MΩ)
Input 1	500.00	-795.46	4.23	0.30
Input 2	500.00	-101.46	0.92	0.82
Control 1	-826.90	20.07	0.35	17.53
Control 2	-838.80	836.29	4.42	0.71
Control 3	77.90	48.01	0.35	2.70
Control 4	423.40	-173.57	1.19	0.64
Control 5	-46.60	161.28	1.10	1.20
Output	0.00	0.02	0.01	231.38
Power		1.24 μW		

Figure 4.12: Bandwidth and energy efficiency scaling. a, The scaling of allowed bandwidth with signal intensity in a log–log plot. The back, blue and red solid lines represent three different indicated cases. Larger required SNR (red) and smaller R_{IV} (blue) lower the bandwidth. The horizontal black dashed line represents the limit set by the hopping relaxation time at 77 K, which increases with temperature. b, The scaling of equivalent energy efficiency with signal intensity in a log–log plot. Larger SNR (red) and smaller R_{IV} (blue) lowers the energy efficiency. The horizontal black dashed line represents the limit at 77 K and fixed power consumption. If the dopant network power consumption is lowered, then the limit and all three scaling trends shift upwards. The three black dotted lines mark three representative computational technologies, the most energy efficient high-performance computer[46], the neural network (NN) accelerator[29] and memristors[47] (section 4.7.8). c, Current flow pattern of a NAND gate (NAND10 in d) with inputs 500 and 0 mV. There is a large parasitic current flowing from input 1 to control electrode 2 (black curved arrow). This parasitic current limits the energy efficiency. This can be solved by using electrostatically coupled electrodes (section 4.7.8). d, Measured power consumption of a NAND gate for the four input combinations. The standard deviations in the current are calculated from ten measurements. The differential resistances R_{diff} are measured around the voltages in the second column.

4.7.9 Feature Mapping and MNIST Classification

Nonlinear mapping of data to a higher-dimensional space[4], and linear separation of the transformed data in this higher-dimensional space are two important aspects for efficient classification[49]. In physical reservoir computing, nonlinear temporal behaviour of a system is needed[5, 8], together with sequential coding in the time domain, to map data and to obtain the sparsity and separation required by the Cover's Theorem[4]. Whereas in our approach, the separation is achieved by the filter's selection of a specific feature. For instance, the "1011" feature will be separated from the other features along the 12^{th} dimension, because the 12^{th} filter yields the largest output current when the "1011" feature is presented (see Figure 4.4a).

To confirm that data separation by the evolved filters is a useful approach independent of the output current scales, we have permuted the output characteristics of filter 1011 to create another "virtual" set of filters, by shifting circularly the output current in Figure 4.4a leftwards by 1 column to create virtual filter 1010, 2 columns to create virtual filter 1001, and so on, to create the other filters. This virtual set of filters also led to over 96% classification accuracy. This result shows that the enhancement of the accuracy of a linear classifier is not affected by the output current scales, but is only caused by the separability brought about by the nonlinear mapping of the data to a high-dimensional space. This finding is well known in machine learning[4], where even random nonlinear mappings to a high-dimensional feature space enhance linear separability of data. We note that we do have to evolve the filters to achieve separation. If we set the control voltages randomly, the output currents are mostly zero or clipped to compliance, and separation is not possible. As mentioned in the main text, a linear classifier is immune to device-to-device variations. As long as there exist (hyper-)planes that separate the classes of data, the weights in the linear classifier are tuned in the training phase to adapt to the different output current scales. To demonstrate the robustness of our approach against imperfect filters, we performed classification with another set of filters evolved in a different device from the one shown in Figure 4.13, in which fifteen filters successfully filter the targeted features and one does not. The accuracy still remains 96%, which suggests that the total classification procedure is tolerant to at least one imperfect filter. These results lead us to the conclusion that, on the system level, as long as the devices filter their target features sufficiently well, overall classification is robust.

The tolerance of our approach to device defects and variations can be further understood by comparing to other approaches to classify the MNIST handwritten digits. For instance, dynamical memristors[8], working as physical reservoirs sequentially processing data coded in the time domain, can also facilitate linear classification even when the actual values of the transformed data are random. Furthermore, the so-called extreme learning machines (ELMs), which are one-hidden-layer, feed-forward artificial neural networks with randomly chosen, fixed weights, can boost the accuracy of MNIST digits classification to 98%[50] (with 15,680 hidden neurons

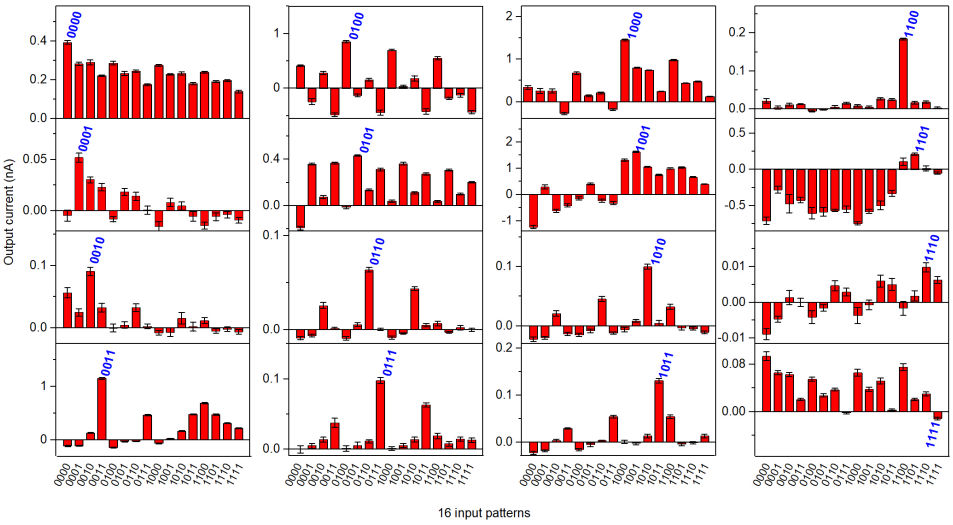


Figure 4.13: Experimental response of the 16 filters. Each of them is evolved to filter the feature given in blue. The output currents corresponding to features other than the desired one are not zero, but the output current of the targeted feature is clearly separated from the other currents. Error bars represent the standard deviation obtained from ten tests.

fully connected to the 784 inputs, in total almost 25 million linear operations and 15,680 nonlinear activations), close to the best deep-neural-network result trained with backpropagation[13]. Therefore, in our case, the actual device output current scales do not matter, as long as the device is evolved to separate data well enough. We like to point out that an ELM with similar accuracy (94.8%) as ours requires 1,568 hidden neurons[50], meaning almost 2,500,000 linear operations and 1,568 nonlinear activations. Performing all these operations on a traditional computer consumes much more energy than our approach.

Let us consider a rough estimation of the energy consumption of this latter, 1,568 hidden neuron ELM running on the top Green500 computer. To evaluate the output of the hidden layer, we need to perform 2.5 million operations at around 60 pJ/OP[46]. This amounts to an energy consumption of 150 μJ, disregarding the computational cost of the nonlinear operations, which are by no means negligible. In contrast, to obtain a similar performance with 11,664 of our devices (required for the MNIST classification), we have a total energy consumption of 11,664·50 $ns \cdot 1\ \mu W = 0.58\ nJ$ per evaluation (50 ns corresponds to the signal reading time with 100 MHz bandwidth, see section 4.7.8). We remark that here we consider the power consumption of the whole device without any assumptions about the amount of operations performed by our device. Furthermore, a small ANN trained with backpropagation should have at

least 15 hidden neurons to achieve comparable accuracy[51]. It requires more than 23,500 operations to activate all 15 hidden neurons, which amounts to an energy consumption of 2.3 nJ on state-of-the-art 4-bit NN accelerators[29], 5 times more than our solution.

The comparison with ELMs and deep neural networks suggests that we can improve accuracy and reduce the number of devices by leveraging deep architectures. We are very much aware that an ANN with deep architectures running on a single traditional personal computer can classify the MNIST digits with over 99% accuracy[13]. One should bear in mind that this accuracy is the culmination of over half a century of development in traditional computing hardware and software.

In real applications, the output of the filters will be superimposed with noise, mainly from the resistor R_{IV} (Figure 4.11c, see also section 4.7.8). The weight matrix trained under noise-free condition tends to be sensitive to additive noise, especially when the signal-to-noise ratio is limited (due to limited number of control electrodes, presently 3). As adding noise in training helps to boost the noise resilience[52] due to its regularisation effect, we trained the weight matrix 10 times with random white noise (0.001 nA amplitude) added to the output of the 11,664 devices for all 60,000 training digits. The average weight matrix of the 10 separate trainings is then used to test its resilience to noise from R_{IV}, which is again superimposed on the 11,664 outputs

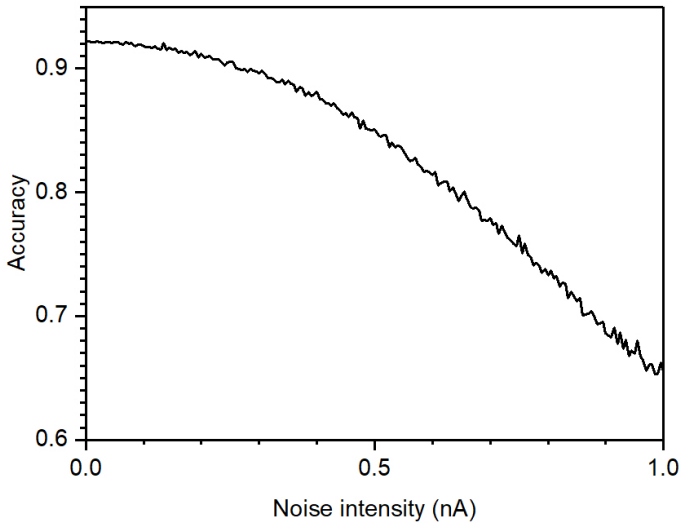


Figure 4.14: Enhancing robustness of the linear classifier against noise. Besides optimizing the SNR, the linear classifier's tolerance to noise can also be increased by taking noise into account during the training phase. The accuracy remains over 92% at 0.05 nA noise amplitude (see section 4.7.8 for a detailed discussion).

from the feature mapping layer for the 10,000 test digits. Figure 4.13 shows that the accuracy at zero noise is about 92.2%, but remains at 92% even with up to 0.05 nA noise, 5 times larger than the measured noise intensity of about 0.01 nA (see Figure 4.13). This robust accuracy is still comparable to state-of-the-art results in other materials-based systems like dynamical memristors[8] or optical networks[7]. In contrast to physical reservoir computers, which take milliseconds to obtain results[8], our system involves no time dynamics and can thus process data in parallel on the order of nanoseconds with high throughput (see above). Compared with optical diffractive networks, which also allow parallel data processing[7], our dopant network has a much smaller footprint (~ 100 nm vs centimetres).

References

[1] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148, 1959. doi: 10.1113/jphysiol.1959.sp006308.

[2] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521, 2015. doi: 10.1038/nature14539.

[3] S. Haykin. Neural networks and learning machines. *Pearson Prentice Hall*, 2008.

[4] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. ieee trans. electron. *Comput. EC*, 14, 1965. doi: 10.1109/PGEC.1965.264137.

[5] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, and J. Grollier. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*, 547, 2017. doi: 10.1038/nature23011.

[6] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: a review. *Neural Netw.*, 115, 2019. doi: 10.1016/j.neunet.2019.03.005.

[7] X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, and A. Ozcan. All-optical machine learning using diffractive deep neural networks. *Science*, 361, 2018. doi: 10.1126/science.aat8084.

[8] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu. Reservoir computing using dynamic memristors for temporal information processing. *Nat. Commun.*, 8, 2017. doi: 10.1038/s41467-017-02337-y.

[9] C. S. Hung and J. R. Gliessman. Resistivity and hall effect of germanium at low temperatures. *Phys. Rev.*, 96, 1954. doi: 10.1103/PhysRev.96.1226.

[10] N. F. Mott. Conduction in glasses containing transition metal ions. *J. Non Cryst. Solids*, 1, 1968. doi: 10.1016/0022-3093(68)90002-1.

[11] V. F. Gantmakher. Electrons and disorder in solids. *Clarendon Press*, 2005. doi: 10.1093/acprof:oso/9780198567561.001.0001.

[12] S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nat. Nanotechnol.*, 10, 2015. doi: 10.1038/nnano.2015.207.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86, 1998. doi: 10.1109/5.726791.

[14] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi. Scaling for edge inference of deep neural networks. *Nat. Electron.*, 1, 2018. doi: 10.1038/s41928-018-0059-3.

[15] A. G. Zabrodskii and K. N. Zinov'eva. Low-temperature conductivity and metal-insulator transition in compensate n-ge. *Sov. Phys. JETP*, 59, 1984.

[16] M. Jenderka, J. Barzola-Quiquia, Z. Zhang, H. Frenzel, M. Grundmann, and M. Lorenz. Mott variable-range hopping and weak antilocalization effect in heteroepitaxial na2iro2 thin films. *Phys. Rev. B*, 88, 2013. doi: 10.1103/PhysRevB.88.045111.

[17] J.F. Miller and K. Downing. Evolution in materio: looking beyond the silicon box. In *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, pages 167–176, 2002. doi: 10.1109/EH.2002.1029882.

[18] S. Harding and J.F. Miller. Evolution in materio: a tone discriminator in liquid crystal. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1800–1807 Vol.2, 2004. doi: 10.1109/CEC.2004.1331114.

[19] M. Mohid and J. Miller. Evolving robot controllers using carbon nanotubes. In *Proceedings of the European Conference on Artificial Life*, pages 106–113, 07 2015. doi: 10.7551/978-0-262-33027-5-ch025.

[20] S. Wolfram. Approaches to complexity engineering. *Physica D*, 22, 1986. doi: 10.1016/0167-2789(86)90309-X.

[21] J. Backus. Can programming be liberated from the von neumann style? a functional style and its algebra of programs. *Commun. ACM*, 21, 1978. doi: 10.1145/359576.359579.

[22] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.*, 14, 2002. doi: 10.1162/089976602760407955.

[23] M. Dale, S. Stepney, J. Miller, and Martin Trefzer. Reservoir computing in materio: An evaluation of configuration through evolution. 12 2016. doi: 10.1109/SSCI.2016.7850170.

[24] M. T. Björk, H. Schmid, J. Knoch, H. Riel, and W. Riess. Donor deactivation in silicon nanostructures. *Nat. Nanotechnol.*, 4, 2009. doi: 10.1038/nnano.2008.400.

[25] M. Pierre, R. Wacquez, X. Jehl, M. Sanquer, M. Vinet, and O. Cueto. Single-donor ionization energies in a nanoscale cmos channel. *Nat. Nanotechnol.*, 5, 2010. doi: 10.1038/nnano.2009.373.

[26] A. Hartstein and A. B. Fowler. High temperature 'variable range hopping' conductivity in silicon inversion layers. *J. Phys. C*, 8, 1975. doi: 10.1088/0022-3719/8/11/007.

[27] M. Minsky and S. A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 09 2017. doi: 10.7551/mitpress/11301.001.0001.

[28] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGPLAN Not.*, 49, 2014. doi: 10.1145/2692915.2628161.

[29] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H-J. Yoo. Unpu: an energy-efficient deep neural network accelerator with fully variable weight bit precision. *IEEE J. Solid-State Circuits*, 54, 2019. doi: 10.1109/JSSC.2018.2865489.

[30] H. Tertilt, J. Bakker, M. Becker, B. Wilde, I. Klanberg, B. Geurts, W. G. van der Wiel, A. Heuer, and P. A. Bobbert. Mechanism for reconfigurable logic in disordered dopant networks. 07 2021. doi: 10.21203/rs.3.rs-757616/v1.

[31] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. Graves, Z. Li, J. W. Strachan, P. Lin, Z. Wang, M. Barnell, Q. wu, S.n Williams, J. Joshua Yang, and Q. Xia. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.*, 1, 2018. doi: 10.1038/s41928-017-0002-z.

[32] J. Tapson and A. Schaik. Learning the pseudoinverse solution to network weights. *Neural Netw.*, 45, 2013. doi: 10.1016/j.neunet.2013.02.008.

[33] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. 2018. doi: 10.48550/arXiv.1712.06567.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014. doi: 10.48550/ARXIV.1412.6980.

[35] A. Aharony, Y. Zhang, and M. P. Sarachik. Universal crossover in variable range hopping with coulomb interactions. *Phys. Rev. Lett.*, 68, 1992. doi: 10.1103/PhysRevLett.68.3900.

[36] J. Pettersson, P. Wahlgren, P. Delsing, D. B. Haviland, T. Claeson, N. Rorsman, and H. Zirath. Extending the high-frequency limit of a single-electron transistor by on-chip impedance transformation. *Phys. Rev. B*, 53, 1996. doi: 10.1103/PhysRevB.53.R13272.

[37] H. Angermann, Th. Dittrich, and H. Flietner. Investigation of native-oxide growth on hf-treated si(111) surfaces by measuring the surface-state distribution. *Applied Physics A*, 59(2):193–197, Aug 1994. doi: 10.1007/BF00332216.

[38] C.N. Berglund. Surface states at steam-grown silicon-silicon dioxide interfaces. *IEEE Transactions on Electron Devices*, ED-13(10):701–705, 1966. doi: 10.1109/T-ED.1966.15827.

[39] L. M. Terman. An investigation of surface states at a silicon/silicon oxide interface employing metal-oxide-silicon diodes. *Solid State Electronics*, 5(5):285–299, October 1962. doi: 10.1016/0038-1101(62)90111-9.

[40] G. K. van Ancum, M. A. J. Verhoeven, D. H. A. Blank, and H. Rogalla. Electric-field activated variable-range hopping transport in prba2cu3o7-. *Physical review B: Covering condensed matter and materials physics*, 52(8):5598–5602, 1995. doi: 10.1103/PhysRevB.52.5598.

[41] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4035–4043. PMLR, 06–11 Aug 2017.

[42] B. I. Shklovskii. $1/f$ noise in variable range hopping conduction. *Phys. Rev. B*, 67:045201, Jan 2003. doi: 10.1103/PhysRevB.67.045201.

[43] A. Agarwal and J. H. Lang. *Foundations of Analog & Digital Electronic Circuits*. Elsevier, 2005.

[44] A. Hunt. The a.c. conductivity of variable range hopping systems such as amorphous semiconductors. *Philosophical Magazine B*, 64(5):579–589, 1991. doi: 10.1080/13642819108217882.

[45] K. Leboeuf, A. H. Namin, R. Muscedere, H. Wu, and M. Ahmadi. High speed vlsi implementation of the hyperbolic tangent sigmoid function. *2008 Third International Conference on Convergence and Hybrid Information Technology*, 1:1070–1073, 2008. doi: 10.1109/ICCIT.2008.131.

[46] The green500 top500.orghttps://www.top500.org/green500/ (2019). URL `https://www.top500.org/green500/`.

[47] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, and J. P. Strachan. Memristor-based analog computation and neural network classification with a dot product engine. *Adv. Mater.*, 30, 2018. doi: 10.1002/adma.201705914.

[48] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E Eleftheriou. Compressed sensing recovery using computational memory. *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 28.3.1–28.2.4, 2017.

[49] R. Legenstein and W Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20:323–334, 2007. doi: 10.1016/j.neunet.2007.04.017.

[50] P. de Chazal, J. C. Tapson, and A. van Schaik. A comparison of extreme learning machines and back-propagation trained feed-forward networks processing the mnist database. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2165–2168, 2015. doi: 10.1109/ICASSP.2015.7178354.

[51] M.A. Nielsen. Neural networks and deep learning, 2015. URL `http://www.neuralnetworksanddeeplearning.com/`.

[52] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, January 1995. doi: 10.1162/neco.1995.7.1.108.

5

A Deep-Learning Approach to Realising Functionality in Nanoelectronic Devices

Many nanoscale devices require precise optimisation to function. Tuning them into the desired operation regime becomes increasingly difficult and time-consuming when the number of terminals and couplings grows. Imperfections and device-to-device variations hinder optimisation using physics-based models. Deep neural networks (DNNs) can model various complex physical phenomena but, so far, are mainly employed as predictive tools. Here, we propose a generic deep-learning approach to efficiently optimise complex, multi-terminal nanoelectronic devices for desired functionality. We demonstrate our approach for realising functionality in a disordered network of dopant atoms in silicon. We model the device's input-output characteristics with a DNN, and subsequently optimise control parameters in the DNN model via gradient descent to realise various classification tasks. When the corresponding control settings are applied to the physical device, the same functionality is found as predicted by the DNN model. We expect our approach to contribute to fast, in-situ optimisation of complex (quantum) nanoelectronic devices.

This chapter is published as: H.-C. Ruiz Euler, M. N. Boon, J. T. Wildeboer, B. van de Ven et al. A deep-learning approach to realizing functionality in nanoelectronic devices. Nat. Nanotechnol. **15**, 992–998 (2020). doi: 10.1038/s41565-020-00779-y

Contributions: Device Fabrication, characterisation and proof of concept measurements.

5.1 Introduction

Exploring the behaviour of nanoelectronic devices can be a time-consuming task, requiring considerable human and experimental resources. For instance, multi-terminal nanoelectronic devices for quantum technology[1, 2, 3, 4, 5], and hardware-based computational paradigms[6, 7, 8, 9, 10] require delicate tuning of control voltages to achieve a desired functionality. Tuning by hand becomes increasingly challenging for devices with a growing parameter space. As the number and complexity of interconnected nanoelectronic devices increases, the demand for automated tuning methods rises as well. Existing methods rely either on search heuristics[1, 3, 4] or, increasingly, on machine-learning methods that combine measurements with either image analysis[11, 12, 13, 14, 15] or gradient estimation of the output response[5] to iteratively converge to the desired functionality.

This article proposes the use of a deep neural network[16] (DNN) model of a nanoelectronic device for optimising the values of the device's control settings to achieve a desired functionality. Tuning the corresponding control parameters of the DNN model (also known as a surrogate model) – instead of the control settings of the physical device – has several advantages. These include a significant reduction in tuning time, as well as in human and experimental resources since the control parameters can be tuned in a completely automated manner with minimal physical measurements. The model is created by training a DNN with a measured training data set that represents the input-output characteristics of a multi-terminal nanoelectronic device. DNNs have been shown to act as efficient function approximators[17] of multidimensional functions. Up to now, in physics, DNNs have been introduced mainly as a predictive tool[18, 19, 20, 21, 22]. Here, we demonstrate that we can successfully realise functionality in a nanoelectronic device by optimisation of its DNN model, which acts as a proxy of the physical device, via gradient descent in a fast and fully automated way. Stochastic gradient descent uses a cost function (see section 5.7) that describes the desired behaviour of the model. By tacking the gradient (partial derivative) of this function with regards to the to be optimised parameter, it is possible to adjust the parameter such that the output of the systems moves to one of the minima of the cost function. In conventional DNN approaches, all the weights of the DNN are trained using gradient descent[23, 24, 25]. However, for the DNN model of a nanoelectronic device, the optimisation is performed for the DNN inputs that correspond to the tuneable parameters of the devices, see the squares in Figure 5.1.

Figure 5.1 illustrates our general approach of creating a DNN model of a multi-terminal nanoelectronic device, and the process of obtaining the desired functionality. For the case study of this article, voltages are applied at the input terminals and currents are measured at the output terminals. First, we sample the multidimensional space of input voltages to obtain a sufficiently large amount of input-output data. Next, we set up a DNN architecture with the numbers of inputs and outputs matching those of the device. The DNN should have a sufficiently large number of

hidden layers and neural nodes per layer to accurately describe the input-output data (See section 5.7 and Figure 5.7). Then, the DNN is trained with measured training data and tested with unseen measured data. The realisation of the desired functionality is achieved as follows. We assign some input terminals as 'control' terminals (circles in Figure 5.1). The control voltages applied at these terminals control the input-output characteristics between the output currents and the voltages applied at the input terminals (squares). At this stage, the control voltages that are to be applied to the device become the learnable control parameters of the DNN. A desired functionality, defined as a specific targeted input-output relationship, is then searched for in the DNN model using gradient descent on the control parameters. Finally, the obtained functionality is verified by applying the obtained corresponding control voltages directly to the physical device, without the need for any further experimental optimisation.

We demonstrate this approach for a multi-terminal nanoelectronic device recently investigated by us[6]. The device consists of an electrically tuneable network of boron dopants in silicon (Si:B) with 8 terminals (electrodes), 7 acting as voltage inputs and one as current output. The active area of the device has a diameter of about 300 nm. The device can be tuned to solve two-dimensional categorical nonlinear binary classification problems[26] using an evolutionary approach[27]. Boolean functionality is realised by applying four input voltage combinations to two input terminals, corresponding to the data inputs 00, 01, 10, and 11, and tuning the remaining terminal voltages such that the four data combinations are mapped to the desired logic gate represented by its output current levels. Figure 5.2a shows typical IV-characteristics measured at 77 K, demonstrating the nonlinear dependence of the output current I_{out} at terminal 8 as a function of the voltage applied at each of the other 7 terminals, while grounding the remaining terminals.

5.2 Deep Neural Network Modelling

To train a DNN model of the device, the 7-dimensional space of its input voltages must be sampled and the corresponding output current measured. One way to obtain these input-output data would be to sample the input voltage space uniformly and randomly, which would be optimal when no information on the input-output relation is available. However, the associated abrupt voltage jumps cause a transient response, viz. a time-dependent current related to capacitive effects. To circumvent this, pauses of at least 20 ms ($\sim$5 times the RC time of the device in combination with the measurement circuit) after input voltage steps could be incorporated to let the system settle into a steady state. As the amount of voltage combinations grows exponentially with the number of terminals, this rapidly becomes very time-consuming.

To solve this problem, we sample the space of input voltages using sinusoidal or triangular modulation functions and a sampling frequency of 50 Hz (see Figure 5.2b and

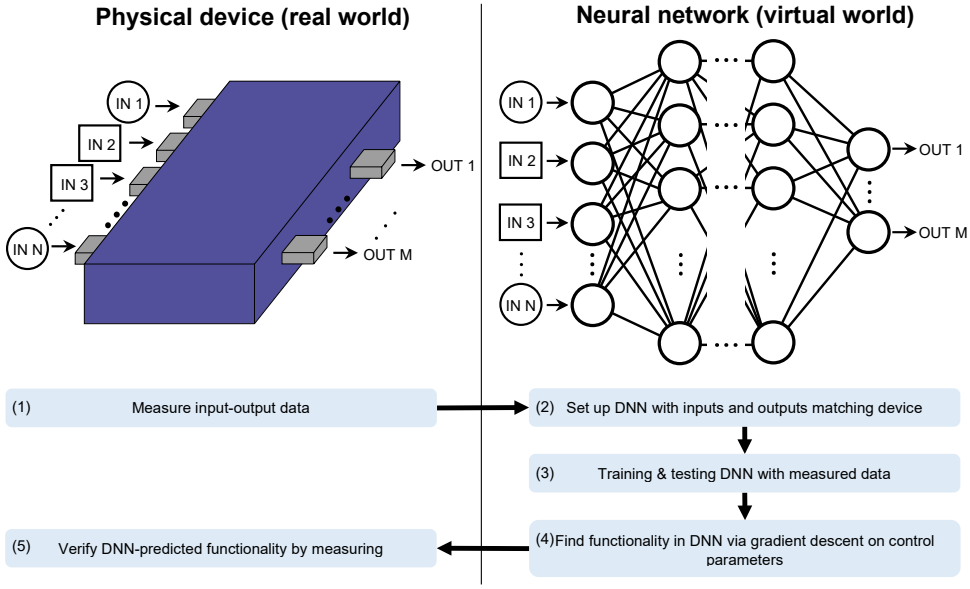


Figure 5.1: Realising functionality in a nanoelectronic device using a DNN model. The following steps are followed to create a DNN model that captures the input-output characteristics of a multi-terminal nanoelectronic device with N input and M output terminals, and to realise a desired functionality. First, input-output data of the device are measured (1). Next, a sufficiently deep and wide DNN is set up, with numbers of inputs and outputs matching those of the device (2). The DNN is trained with the measured training data and tested with unseen test data (3). The DNN is used to find the desired functionality using gradient descent on the control parameters (4). The predicted corresponding control voltages are then applied to the physical device to verify the functionality (5). Circles and squares pointing towards the device/DNN indicate the control and data inputs, respectively. Circles in the DNN represent artificial neurons and their activation function.

section 5.7). This sampling technique minimises discontinuities in the applied voltages and therefore reduces transient effects. The efficiency of sampling this way depends strongly on the choice of the modulation frequencies. By choosing the modulation frequencies of the different input voltages such that the ratios of all frequency combinations are irrational (see Table 5.1), we guarantee that the input space is densely covered and that no recurrences of voltage combinations occur. Also, the sampling density can then be increased by simply increasing the sampling time. Our approach has significant advantages over standard sampling with a predefined uniform grid in the 7-dimensional voltage space. For the same total number of samples and highest modulation frequency as in our approach, which is limited by remaining transient

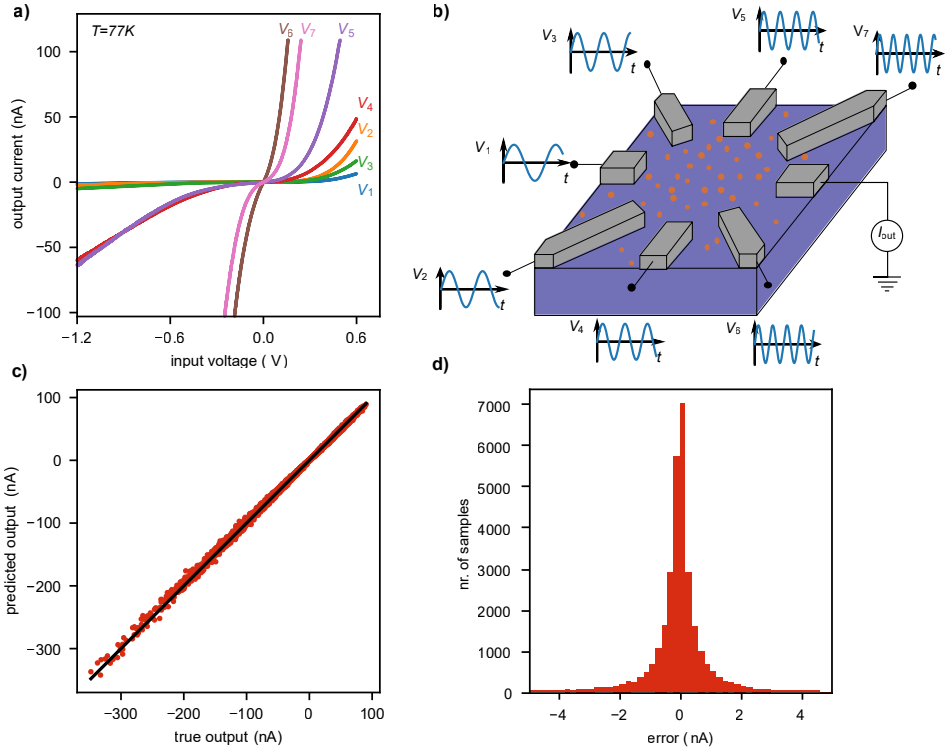


Figure 5.2: Sampling input-output data to train and test the DNN. (a) Typical IV-characteristics of the Si:B device measured at 77 K, showing nonlinear behaviour. The current between the output terminal and ground is measured while applying a voltage to each of the input terminals and grounding the remaining terminals. (b) Schematic representation of the device, with boron dopants represented by orange dots (see Ref. [6] for details). Training and test data are obtained by measuring the output current (I_{out}), while applying sinusoidal (as shown in this example) or triangular voltage modulations (V_1 -V_7) to the input terminals (frequencies given in Table 5.1). (c) Output current predicted by the trained DNN for the unseen test data set against the current measured in the physical device. The solid line has slope 1. (d) Histogram of the test error, showing an RMSE of 1.2 nA.

effects, the total sampling time in the standard approach would need to be around 5 times longer (see Table 5.2). In addition, an increase of the sampling density would require performing a new sampling over a new grid instead of simply increasing the sampling time.

The input data consist of tuples of input voltages (V_1-V_7, in V) and the output data are scalars representing the output currents (I_{out}, in nA). The input and output layers of the DNN correspond to the input terminals and output terminal of the device, respectively. We model our nanoelectronic device with a fully connected DNN consisting of 7 inputs and 1 output. A network architecture with 5 hidden layers and 90 nodes per layer is found to have a sufficiently small test error (see Figure 5.7). Minimisation of the mean-squared error is used for training the DNN (see section 5.7).

Figure 5.2c shows the output current predicted by the DNN vs the measured output current for unseen measured test data. We observe only small deviations from the identity curve (black), as compared to the overall range of the output. The root-mean-square error (RMSE) in the predicted currents for the test data is 1.2 nA, see Figure 5.2d, which is 0.27% of the total current output range. These results show that the trained DNN predicts the unseen data accurately.

5.3 Automatic Functionality Search via Gradient Descent

A desired functionality is specified by a targeted dependence of the output current on one or more input voltages of selected terminals. The functionality is obtained by learning the values of the remaining input voltages, i.e. the control parameters, by following the negative gradient of a cost function E(y,z), which is a measure of the similarity between the predicted outcome data y and the targeted outcome data z. At this stage, the internal weights of the DNN are kept frozen. We use a cost function composed of a correlation factor and the logistic function (see section 5.7). In contrast to the mean-squared error, this cost function does not target specific output current values, but rather promotes separation of low ('0') and high ('1') output currents. The entire process of automated optimisation by the DNN model is represented in Figure 5.3 for the case of an XOR Boolean logic gate. To demonstrate the speed and accuracy of this approach, we apply it to solve different tasks with increasing accuracy requirements.

5.4 Classification with DNN-optimised Nanoelectronic Devices

Starting with Boolean logic gates, voltages V_2 and V_3 on terminals 2 and 3 are used as data inputs, as identified by the squares in Figure 5.3a and the yellow and blue coloured terminals in Figure 5.3b. Hence, there are five remaining control voltages (V_1, V_4, V_5, V_6, V_7) to realise the gates. We note that it is to a large extent a matter of choice which terminals are used for data input and which for control.

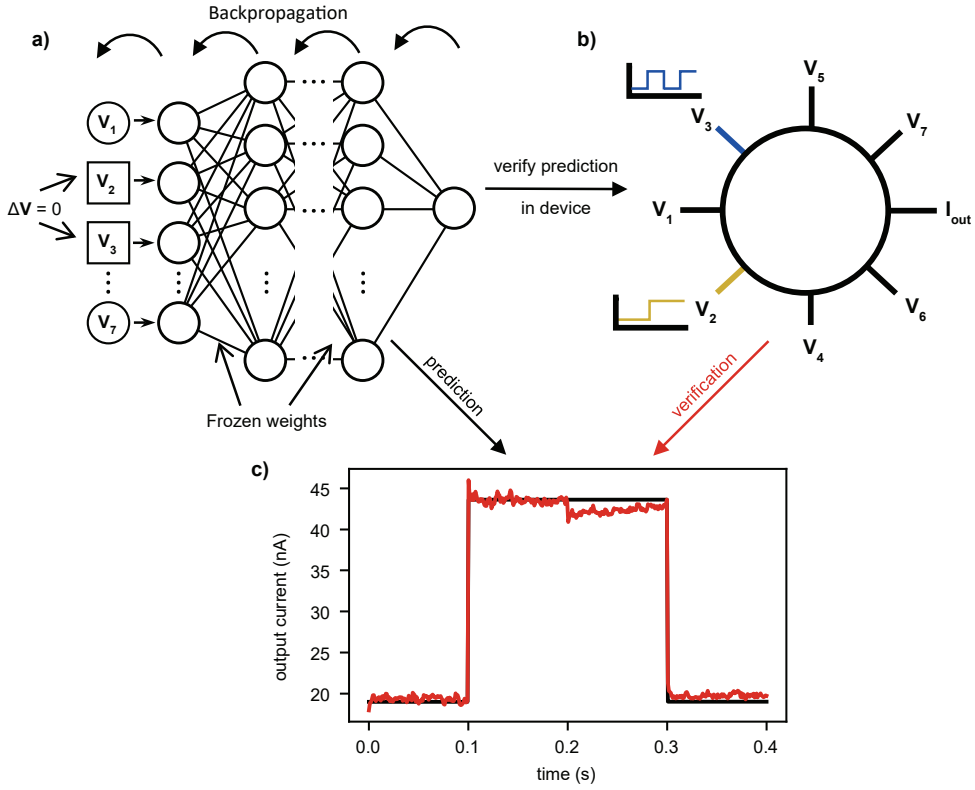


Figure 5.3: DNN prediction of device functionality and verification. (a) Backpropagation of the gradient through the DNN to tune the control parameters realising the desired functionality. During the tuning process, the internal weights of the DNN are frozen. The input and control terminals are represented as squares and circles, respectively. (b) For this example, the prediction of an XOR Boolean logic gate is chosen, consisting of 2-dimensional input data (V_2 and V_3, $-1.2\ V$ for '0' and $0.6\ V$ for '1'), with the 5 remaining voltages as control parameters. The control voltages are tuned such that the input data are mapped to the desired outputs. (c) To verify the predicted outcome, the tuned control voltages and the time-dependent input voltages are applied to the physical device. Each input combination is applied for 0.1 s, resulting in a total output signal of 0.4 s, which is sufficiently long to reveal typical fluctuations in the output current. The predicted outcome is shown in black and the physical measurement in red. In between the different input combinations, the input voltages are linearly ramped in 10 ms to their new values, during which the current is masked. We note that optimisation of the devices and the interfacing equipment may eventually lead to a readout bandwidth exceeding 100 MHz[6].

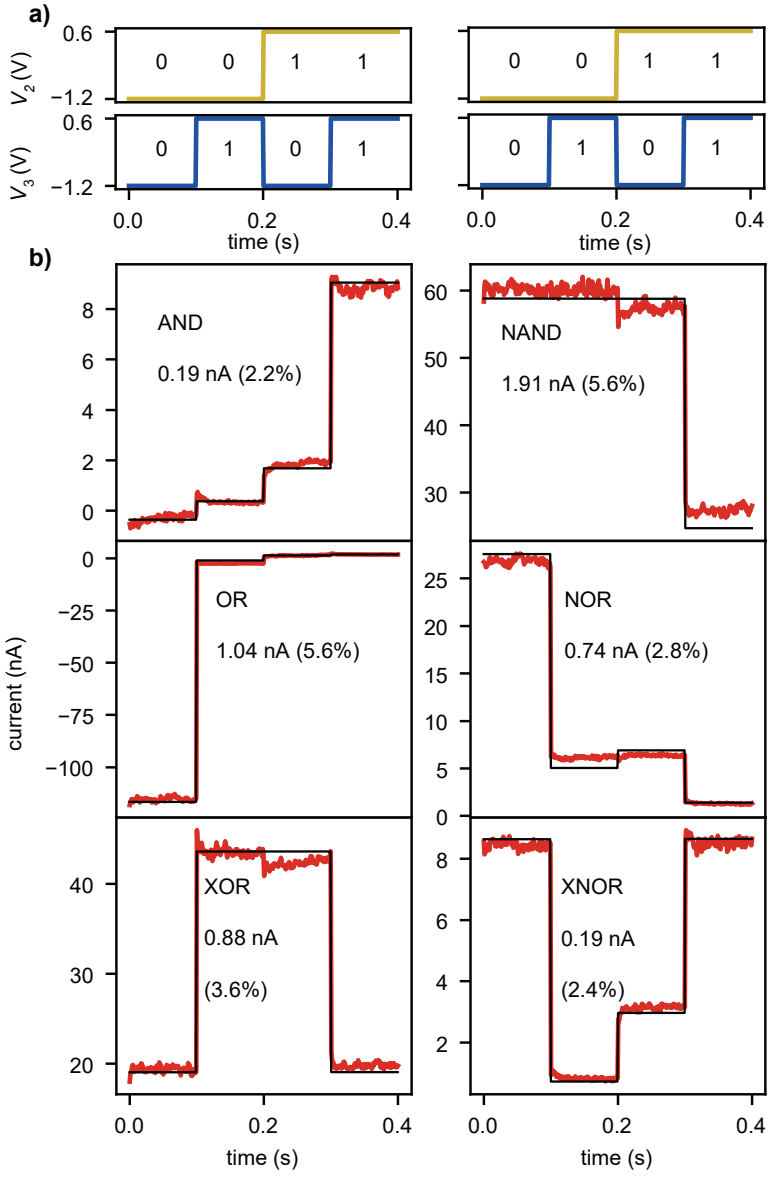


Figure 5.4: Prediction and verification of Boolean logic. (a) Applied voltages V_2 and V_3 to test the Boolean logic gates. (b) Black curves: logic gates predicted by the trained DNN. Red curves: output current measured in the device using the control voltages predicted by the DNN (given in Table 5.5). In between the different inputs the voltages are linearly ramped in 10 ms to their new values, during which the current is masked. The numbers in the panels indicate the RMSE (nA) and the normalised RMSE (in brackets), respectively. The horizontal dashed lines indicate current levels that could be used for separating the logical outputs '0' and '1'.

The desired functionality (as determined by the accuracy of the benchmark task) is generally obtained when the input terminals are not neighbouring the output terminal or each other, which is intuitively understandable from the underlying physics of variable-range hopping of charge carriers in between the dopants and between the dopants and the electrodes in the electrostatic potential generated by the electrode voltages and the dopants and charge carriers themselves[6]. During verification of the predicted gates the input data (V_2, V_3) are presented to the physical device as a time sequence in the order shown in Figure 5.3b. These input wave forms, given by the yellow (0011) and blue (0101) signals, represent all four possible combinations of inputs for the truth table for Boolean logic (00, 01, 10 and 11). We show in Figure 5.3c the numerical prediction and experimental verification of an XOR gate in black and red, respectively.

Figure 5.4 shows the results for all logic gates. The voltage values corresponding to logic inputs '0' and '1' are $-1.2\ V$ and $0.6\ V$, respectively; see Figure 5.4a. In Figure 5.4b, we show in black solid lines the output currents for the logic gates predicted by the DNN. To verify the predicted output currents, the predicted control voltages and the binary input voltages are applied to the physical device. The measured output currents are shown by red curves. A comparison shows that all gates predicted by the DNN are also demonstrated in the physical device. Moreover, the values of the output currents are predicted with high accuracy. The RMSEs of the predicted gates (numbers in Figure 5.4b) are consistent with the test error of $1.2\ nA$ in Figure 5.2d. The normalised RMSEs (numbers in brackets) display the magnitude of the error with respect to the current scale (highest minus lowest current) of the predicted gates. The normalised RMSEs show that the worst predicted gates (NAND and OR) have a relative error of 5.6%. Possible threshold current values separating 'true' and 'false' for each logic gate are represented by the dashed lines in Figure 5.4b. Optimal thresholds could be automatically determined by training a single perceptron[28] on the output and target data.

The DNN model allows for an accelerated exploration of functionality, without performing further measurements on the device after collecting the training and test data. In our case, we are able to reduce the search of Boolean functionality from 15 minutes per gate, by performing physical measurements in combination with an evolutionary algorithm[6], to around 10 seconds on a standard computer (Processor: Intel® Core™ i5-8250U CPU @ 1.60GHz, 4 cores and 8 GB RAM) by using the DNN model, i.e. a speed-up of nearly two orders of magnitude.

Next, we study a more challenging binary classification problem, consisting of classifying points in a 2-dimensional feature space with two classes: data points in an outer ring corresponding to the label '0', and data points in an inner cluster corresponding to the label '1'. The points are shown as discs and crosses, respectively, in Figure 5.5a. The two classes in this ring classification problem must be separated using a closed decision boundary. Using in this case V_4 and V_5 in Figure 5.3b as data inputs, the

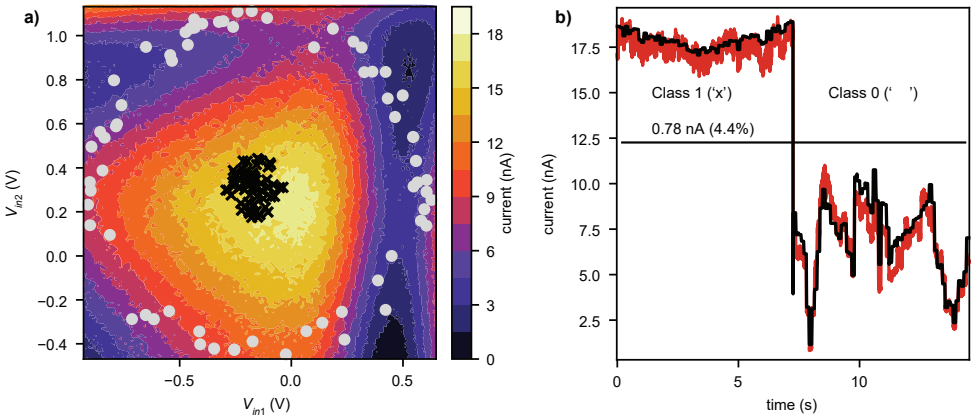


Figure 5.5: Ring classification functionality. (a) 66 outer data points (class '0', grey discs) and 66 inner data points (class '1', black crosses) to be classified. In this case V_4 and V_5 from Figure 5.3b are used as data input voltages $V_{in,1}$ and $V_{in,2}$, while the remaining voltages are the control parameters. The heat map shows the physical device's output current while sweeping V_4 and V_5 and keeping the predicted control voltages fixed (see Table 5.6 for their values). The points undergo an affine transformation before the mapping (see Table 5.7 for the scaling factor and offset voltages). (b) Prediction by the trained DNN (black curve) and verification measurement (red curve). Grouped in classes for visualisation, each data point is measured for 0.1 s in the physical device and in between the data points the input voltages are ramped up/down to the next point with a ramping time of 10 ms, to avoid transient effects. The classes are readily separable by a threshold current (horizontal dashed line).

DNN is trained to separate the two classes such that the inner class corresponds to a high and the outer class to a low output current. The remaining control parameters (V_1, V_2, V_3, V_6 and V_7) are to be tuned. Additionally, we have added a scaling factor and two bias parameters to the input voltages, enabling the DNN to determine an affine transformation of the data by itself (see section 5.7). The resulting prediction of the DNN is shown in Figure 5.5b by the black solid line. The physical measurements, represented by the red curve, take 100 ms per data point. The RMSE is 0.78 nA and the normalised RMSE is 4.4%, demonstrating the accuracy of the prediction by the DNN model. We note that the second input voltage ranges up to 1.1 V, whereas the DNN is only trained for voltages up to 0.6 V for this terminal. The trained DNN thus successfully extrapolates over a 0.5 V range. To visualise the results, Figure 5.5a includes a heat map created by sweeping the two input voltages in the physical device. This demonstrates that the output current indeed maximises for the inner class and minimises for the outer class. The two classes are separable by defining a threshold current as decision boundary; see the dashed line in Figure 5.5b. We note that the used data inputs V_4 and V_5 are different from those for the Boolean gates (V_2

and V_3). A good functionality can also be obtained with other choices for the data inputs. In Figure 5.11 another good ring classification functionality is demonstrated using V_6 and V_7 as data inputs (see Tables 5.3 and 5.4).

5.5 Feature Mapping with DNN-optimised Nanoelectronic Devices

We fabricated another device with the same structure as in Figures. 5.2b and 5.3b of the device used for the previous experiments, and used it to perform a more complex task. The task involves distinguishing 16 different 2×2 pixel features by the device's output current, which is a possible subtask of a higher-level image recognition task[6]. Measurement of training and test data is done as before with, as differences, the use of triangular instead of sinusoidal modulation functions and other voltage amplitudes (see Table 5.1). The modelling of the DNN is the same as before. The RMSE of the predicted test data happens to have the same value of 1.2 nA as before, which is now 0.6% of the output current range. The features are defined by 2×2 black and white pixel combinations that are converted to high (black) and low (white) voltage values on four input terminals, in this case V_2, V_3, V_4 and V_5. The remaining terminal voltages V_1, V_6 and V_7 are the learnable controls. Learnable scaling factors and bias parameters are added to each of the four input voltages. In the search for the desired functionality with the trained DNN, a cost function is used that promotes separation of output currents for the 16 different pixel features (see section 5.7).

Figure 5.6a visualises the main problem faced in the feature mapping. Noise and instabilities of various origin lead to a distribution in the measured output current when a certain pixel feature is presented to the device multiple times. For a reliable mapping, the separation of the output currents for the different features should sufficiently exceed the width of this distribution. Figure 5.6b demonstrates that the functionality obtained from the DNN search satisfies this criterion. In multiple verification measurements the different features are subsequently presented to the device for 0.1 s. The result of a specific measurement is shown in the left panel (red curve) together with the DNN prediction (black solid curve). The right panel shows histograms of the complete set of measurements, involving 1,000 measurement points per feature obtained from a total of 10 measurements at $1,000$ Hz spread over the course of 1 minute. The dashed lines in Figure 5.6b show a set of decision boundaries determined using a naive Bayes classifier that can be used to separate the features (see section 5.7 and Figure 5.12). With these boundaries, 99.94% of the measured data are classified correctly.

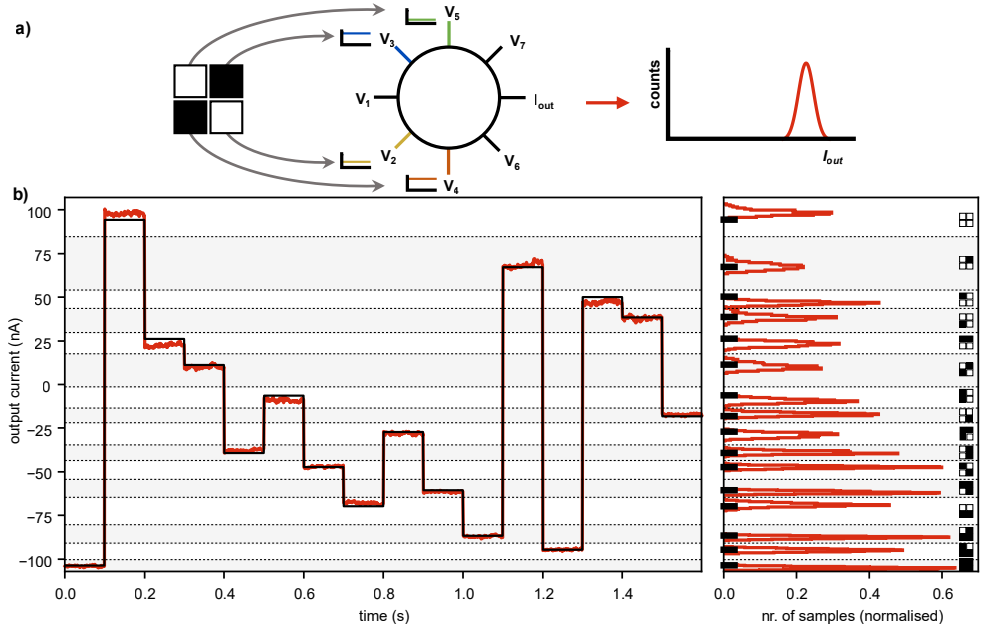


Figure 5.6: Feature mapping task. (a) Schematic representation of the task. All four pixels of the feature are presented as a high (black) or low (white) input voltage to a terminal of the device, after which a histogram of the output current is obtained that should be separated from that of other features. The voltages of the remaining control terminals are optimised by the DNN model to maximise the difference between the output currents of different features (see Supplementary Tables S8 and S9 for the parameters). (b) Left: comparison of a measurement of the output current (red) to the DNN prediction (black) for the different input features. In between different presented features the input voltages are linearly ramped in 200 ms to their new values, during which the current is masked. Right: histograms of the output current of 10 measurements. The dashed lines show decision boundaries obtained with a naive Bayes classifier (see Table 5.10).

5.6 Conclusion

We have proposed a generally applicable deep-learning approach to realising desired functionality in complex multi-terminal nanoelectronic devices. The method involves the training of a deep neural network (DNN) model that emulates the device's multidimensional input-output characteristics, followed by gradient descent on selected control parameters of the DNN to find the desired functionality. The set-up of the DNN model, which involves input-output data collection and training, needs to be done only once per device. After the DNN model set-up, a variety of functionalities

can be searched for with the model without further experimentation on the device. We demonstrated the method for nanoelectronic devices consisting of boron dopants in silicon (Si:B) contacted by 8 terminals. Owing to the high efficiency of input evaluation by a DNN, we were able to find all Boolean functionalities nearly two orders of magnitude faster than by performing physical measurements in combination with an evolutionary algorithm[6]. In addition, we were able to readily realise a more complex ring classification and a 2×2 pixel feature mapping functionality, showing that the devices have more computational capacity than previously shown[6].

We expect our approach to have a broad application range. By using standard machine learning techniques and libraries, the generality of our approach ensures the optimisation of a broad range of physical systems where control parameters are available. Naturally, the choice of DNN architectures determines the reliability of the functionality prediction and varies for different physical systems (see Figures 5.8-5.10 and section 5.7).

As an extension of the present work we intend to explore the coupling of many devices to create hierarchical structures with many more inputs, outputs and control parameters. Finding a desired complex functionality solely by experimentation will then become increasingly challenging. A limitation of our approach will be the time needed to sample the input voltage space, which scales exponentially with the number of input terminals if the sampling density is kept constant. In the present study, the number of input terminals is 7. If the sampling speed can be increased to 100 MHz[6] instead of the present 50 Hz, while keeping the same sampling density, the number of input terminals could be doubled to about 14 without increasing the sampling time. For many more input terminals, a modular approach can be used, where DNN models are created for separate modules with a still manageable number of terminals. The response of the total system can then be modelled by coupling the DNNs of the separate modules.

Our approach can also be valuable for tuning of quantum-dot architectures[29, 30, 31], used in quantum information processing. Large systems consisting of many coupled quantum dots have a large parameter space and therefore require more sophisticated tuning approaches. Existing tuning methods are based on “device-in-the-loop” approaches[32, 33, 34], i.e. measuring the real device is required during optimisation. In contrast, parameter tuning in our approach can be performed entirely off-chip. Because of its generality, our deep-learning approach can be readily applied to these systems. We finally want to mention that our approach is not restricted to nanoelectronic devices, but can be applied to any complex, tuneable, static system, such as programmable metasurfaces[35], for which realising functionality is also challenging.

5.7 Appendix A: Methods

5.7.1 Data Sampling

To perform the measurements, we use a battery-powered electronics rack comprised of voltage sources and IV converters for low-noise measurements. All the measurements are automated with Python using Ni-DAQmx software[36] and our in-house framework SkyNEt[37]. The temperature is fixed by dipping the device in a liquid-nitrogen (77 K) dewar.

The training and test data are sampled using sinusoidal (Boolean logic and ring classification functionality) or triangular (2x2 pixel feature mapping) modulated input signals to minimise capacitive transient effects. We used both input signals to test different input distributions (see Table 5.2). The frequencies, amplitudes, and offsets of these functions for each input terminal are given in Table 5.1. We choose the frequencies proportional to the square root of prime numbers. The ratio of any two frequencies is then irrational, which guarantees a good coverage of the 7-dimensional input voltage space and prevents any recurrence of voltage combinations. To ensure that training and test data do not overlap, the training data set is generated using modulation functions without phase shift, whereas the test set is generated with a phase shift of 1 radian ($\sim$57 degrees). The training and test data sets comprise about 3$\times$106 and 540,000 input-output pairs, respectively. The sampling occurs at a frequency of 50 Hz, where one out of three points is added to the data set. For the generation of the sinusoidal and triangular voltage modulation functions an NI 9264 voltage sourcing module is used in combination with a cDAQ-9171 and cDAQ 9178 chassis. Output currents are measured after an IV conversion with either an NI USB-6216 device or an NI 9202 voltage measuring unit in combination with the cDAQ 9178 chassis.

5.7.2 DNN Architectures

The general methodology to determine the hyper-parameters of DNN models, especially regarding the architecture, is still an open question in the field of deep learning, but suggestions exist to guide the construction of suitable DNN models. If there are no previous examples of network models for the system at hand, it is advisable to start with simple models, e.g. linear models or shallow networks, and assess their ability to predicting the behaviour of the given system. When designing the model, one should consider the amount of data available for the training and validation of the models, for which no clear guidelines exist. However, as a rule of thumb, if simple models readily overfit the data, one should focus more on data acquisition or other modelling methods suitable for small datasets. Given a suitable amount of data, it is best practice to explore architectures with different design choices, such as depth, width, and activation function. At the beginning of the exploration, choosing the width of the hidden layers to be larger than the input dimension often improves performance. Increasing

the depth of the DNN usually boosts performance at the beginning, but once this improvement becomes marginal one should increase the width of the hidden layers. The aim should be to find the architecture with the best performance on validation data that generalises well on test data. An example of this procedure is given in Figure 5.7 in section 5.8.2. If overfitting is observed, one can regularise the DNN to improve performance. If the cost levels off at an acceptable value for different architectures, it is advisable to take the simplest model to avoid computational overhead. If further efficiency considerations are important after finding the best DNN model, one may optimise the computational complexity of the model via parameter quantisation and pruning, low-rank factorisation or knowledge distillation approaches[38]. There are, however, important considerations when making a trade-off between accuracy and computational/model complexity. In general, sacrificing the prediction accuracy will result in bigger efforts in searching for functionality, because the probability of false positives and negatives will increase (see Figures 5.8-5.10). We refer to section 5.8.2 for a detailed discussion on the consequences of this trade-off.

5.7.3 DNN Training

The neural network consists of 7 inputs and 1 output, corresponding to the physical device's input and output terminals, respectively. We use a fully connected feed-forward network, consisting of 5 hidden layers with 90 nodes (see section 5.8.2 for the choice of this architecture) and a ReLU activation function. Training is done by stochastic gradient descent on the mean-squared error for 3,000 epochs with a learning rate of $=10^{-5}$ and a minibatch size of 128. The training data set is split into a set (90%) used for the training and a set (10%) used for the validation. The validation data set should not be confused with the test data set. The validation set is used to prevent overfitting, while the test set of unseen data is used to estimate the generalization error. For optimisation, Adam[39] is used with $\beta_1 = 0.9$ and $\beta_2 = 0.75$. All hyperparameters are explored to obtain the lowest possible test error. Our DNN model is implemented using PyTorch[40].

5.7.4 Cost Functions

The cost function to obtain Boolean logic and ring classification functionality is given by

$$E(y, z) = ((1 - \rho(y, z)))/f((y_{sep} - q)/p) \tag{5.1}$$

where $y = (y_1, y_2, \ldots, y_n)$ are the actual currents, $z = (z_1, z_2, \ldots, z_n)$ the targeted binary current levels (n = 4 for Boolean logic and n = 132 for ring classification), $\rho(y, z)$ is the Pearson correlation coefficient and $f(x) = 1/(1+e^x)$ the standard logistic function. The current values q and p control the desired separation and are chosen as 3 and 5 nA, respectively. The value y_{sep} represents the minimum separation between the high and low labelled data. For the data labelled as class '1' (high output) the lowest predicted current is taken and for the data labelled as class '0' (low output)

the highest predicted current is taken, which are subtracted to obtain y_{sep}. While the correlation function promotes similarity between the targeted and actual outputs, the logistic function promotes separation of the two classes.

Although the binary cross entropy loss is most often used for binary classification tasks, its use would require mapping the output of our device to the posterior probability over the targets. This requires a linear readout to be learned together with the control voltages. Here, we opt for a cost function that avoids introduction of extra parameters. We designed our cost function to be a differentiable function of the control parameters that reflects the characteristics of the fitness function used previously to find Boolean functionality[6]. For the 2×2 pixel feature mapping we use the following cost function,

$$E(y) = -\sum_i f(|y_i - y_{NN(i)}|/p) \tag{5.2}$$

where, y_i is the current for feature $i (i = 1, \ldots, 16)$ and $y_{NN(i)}$ is the 'nearest-neighbour' current ($NN(i)$ is the feature with current closest to y_i). The logistic function promotes an initial increase in current separation and leads to a saturation for a sufficiently large separation. We take $p = 2\ nA$, leading to saturation of the cost function for separations above 10 nA.

5.7.5 Control Parameters

The Boolean logic gates consist of 4 distinct combinations of 2 binary states for the input terminals. The voltages representing these states are taken to be -1.2 V and 0.6 V, corresponding to '0' and '1' respectively. The voltages of the remaining 5 terminals are the learnable control parameters. The input data set of the Boolean logic gates is expanded such that each input combination is represented 100 times, thus leading to a total of 400 data points in the training set. During optimisation, these data points are randomly presented as inputs to the DNN. Stochastic gradient descent is used with a minibatch of 100 data points and a learning rate of =0.08. A single optimisation session for a logic gate consists of at most 600 epochs, with an early termination if in the previous 150 iterations there has not been a significant reduction in error. To prevent the learnable parameters from deviating too much from the voltage range of the training data set, the parameters are regularised with an L1-norm outside this range. To obtain the best results, we re-initialised training 10 times per logic gate. The predicted control voltage values with the lowest error are the ones taken in the verification. These are given in Table 5.5.

For the ring classification problem 132 input data points are used, which are shown in Figure 5.5a. The hyperparameters used for optimisation are the same as for the Boolean logic gate optimisation, except for the number of initialisations and maximum amount of iterations per initialisation, which are 20 and 800, respectively. The 66 outer points are generated uniformly and randomly in a ring between circles with

radii 0.5 and 0.6, and the 66 inner points in a circle with radius 0.1. The coordinates (x_1, x_2) of these points are transformed to input voltages by

$$V_{in,i} = x_i \cdot V_{scaling} + V_{offset,i} \tag{5.3}$$

where $V_{scaling}$ is a scaling voltage, taken equal for the two input electrodes, and $V_{offset,i}$ are voltage offsets. These parameters define an affine transformation of the data and are next to the control voltages the learnable control parameters. The control voltages for the ring classification functionality are given in Table 5.6 and the scaling voltage and offsets of the two input voltages in Table 5.7. Note that for this example the input data terminals are 4 and 5, which are different from the Boolean logic gates (terminals 2 and 3).

For the feature mapping functionality shown in Figure 5.6 we map the 16 combinations of 4-dimensional binary patterns to 16 distinguishable current output levels. The binary values $x_{base} = 0$ for 'high' and 1 for 'low' are mapped to input voltages

$$V_{in,i} = x_{base} \cdot V_{scaling,i} + V_{offset,i} \tag{5.4}$$

where again $V_{scaling,i}$, now taken different for the 4 input electrodes, and $V_{offset,i}$ are added to the learnable control parameters. The final functionality is the best result obtained after 500 random initialisations of the control parameters and training for 5,000 epochs with a minibatch of 4. The resulting control voltages are given in Table 5.8, and the scaling factors and offsets of the four input voltages in Table 5.9.

5.7.6 Pixel Feature Decision Boundaries

The set of decision boundaries for the pixel features shown in Figure 5.6b is determined using a naive Bayes classifier with Gaussian data approximation, which optimises the posterior probability. The collective measured data in the right panel of Figure 5.6b are used as training set for finding the decision boundaries. Using standard Bayesian notation, a new datapoint x is assigned to class k' for which

$$P(k') \cdot P(x|k') = \max_k [P(k) \cdot P(x|k)] \tag{5.5}$$

Here, $P(k)$ is the prior probability, which is equal to the fraction of datapoints belonging to class k in the training set, and $P(xk)$ is the probability density function of class k in the training set. In our case the classes have an equal prior probability, so $P(k) = P_0 \forall k$. The probability density function $P(xk)$ of each class $k = 1, \ldots, 16$ in the training set is in our case assumed to be Gaussian (see the histograms in the right panel of Figure 5.6b). Thus, in our case a new datapoint x is assigned to the class k that has the highest probability density for that value of x. Graphically, this means that the decision boundaries are located at the crossings of Gaussian fits to the histograms in the right panel of Figure 5.6b, as is shown in Figure 5.12. The currents corresponding to the decision boundaries are given in Table 5.10. The classification is done with the Gaussian naive Bayes module of the scikit-learn package[41].

5.8 Appendix B: Supplementary

5.8.1 Comparison of Different Sampling Methods

We compare here the sampling density of three different sampling methods of the 7-dimensional input voltage space of the device: 1) a uniform grid sampling with the fastest running input voltage (i.e. the voltage at an input electrode that swipes the fastest through the defined grid), a one-but-fastest running input voltage, etc., [2]) a sinusoidal sampling with modulation frequencies with irrational ratios given in Table 5.1, and 5.3) a triangular wave sampling with the same modulation frequencies used in [2]. We obtain an estimate of the sampling density and time as follows. The 7-dimensional input voltage space is subdivided into 12 equally sized small bins per input terminal between the lowest and highest voltage, totalling 12^7 bins. For all three sampling methods, the bins containing at least one input data point are regarded as 'sampled' and the empty bins as 'unsampled'. Table 5.2 shows for the three different sampling methods the fraction of sampled bins in the 7-dimensional input space as 'density' and the sampling time (rounded to hours) as 'time', for different numbers of grid points per input terminal. To estimate the sampling time for the uniform grid, the fastest running voltage of the grid sampling is taken to have the same frequency as the highest modulation frequency of the sinusoidal and triangular wave sampling, which is 0.22 Hz (see Table 5.1). For N sample voltages per terminal, the inverse modulation frequencies of the 7 different voltage signals are proportional to $N, N^2, N^3, \ldots, N^7$ and the total sampling time in seconds is then given by $N^7/(2N0.22)$. Table 5.2 shows only small differences in the sampling density for the different sampling methods. However, the sinusoidal and triangular wave sampling methods are considerably faster than the uniform grid sampling. The triangular wave sampling becomes slightly faster than the sinusoidal sampling with growing N.

Table 5.1: Control voltages for the alternative ring classification. Control voltages predicted by the DNN, applied to the device to verify the alternative ring classification in Figure 5.11.

Terminal	1	2	3	4	5	6	7
Frequency / 0.05 (Hz)	$\sqrt{2}$	$\sqrt{3}$	$\sqrt{5}$	$\sqrt{7}$	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{19}$
Amplitude (V), device 1	0.9	0.9	0.9	0.9	0.9	0.5	0.5
Offset (V), device 1	-0.3	-0.3	-0.3	-0.3	-0.3	-0.2	-0.2
Amplitude (V), device 2	0.85	0.85	0.85	0.75	0.75	0.1	0.15
Offset (V), device 1	-0.35	-0.35	-0.35	-0.25	-0.35	0.0	0.0

The two DNN models for the two devices in the main article are trained and tested using data sets obtained during, respectively, 48 hours and 3 hours of sampling with sinusoidal or triangular waves. The sampling in that case is somewhat denser than

Table 5.2: Comparison of different sampling methods. Sampling density (sampled fraction of 12^7 equally sized bins in the 7-dimensional input voltage space) and total sampling time (rounded to hours) for three different sampling methods and different sample voltages N per terminal. The modulation frequencies used in the sinusoidal and triangular wave sampling are given in Table 5.1. A sampling frequency of 50 Hz is used in both methods.

	Uniform grid		Sinusoidal waves		Triangular waves	
N	Density	Time (h)	Density	Time (h)	Density	Time (h)
6	0.00781	30	0.00732	4	0.00738	4
7	0.0230	74	0.0231	13	0.0239	13
8	0.0585	336	0.0592	36	0.0592	33
9	0.133	336	0.135	92	0.135	78
10	0.279	631	0.279	240	0.280	175

N = 8 sample voltages per terminal. Besides the advantage in data acquisition time, sampling with waves reduces the transient behaviour in the output current observed with fast voltage changes.

5.8.2 DNN Architecture

In our optimisation of the deep neural network (DNN) architecture, we consider DNNs with different numbers of hidden layers (1, 2, 5 and 6) and nodes per layer (15, 30, 60 and 90). Figure 5.7 shows the root-mean-square error (RMSE) in the current predicted by the trained DNN as compared to the measured validation data on the physical device for the architectures that we considered. We observe a similar performance for DNNs with 5 and 6 hidden layers, which both show a significantly better performance than DNNs with less hidden layers. Increasing the number of nodes per layer from 60 to 90 decreases the performance of the DNN with 6 hidden layers, possibly due to a slight overfitting of the training data. The best performing DNN is the one with 5 hidden layers and 90 nodes per layer, and we therefore used this DNN in the modelling of our device.

If computational efficiency is a concern, one could trade predictive accuracy (here quantified by the RMSE) for computational efficiency (here quantified by the DNN size). One could, for example, take the smallest architecture for which the RMSE levels off. In our specific case, we could choose a less computationally demanding model with 5 hidden layers and 60 nodes per layer without any significant performance loss (see Figure 5.7). However, in general predictive accuracy should have priority, since failing to predict the device's behaviour correctly significantly increases the probability of false negatives and false positives when searching for functionality.

Figure 5.8 shows the AND gate solved by the device from the main text (red curve, always the same) compared with different current predictions using models with architectures as in Figure 5.7. Although the same input and control voltages are used for all models and the device, we see large variations in the predictions. The simplest models (1L15N, 1L30N and 2L15N) fail to predict the trends in the output current, giving a false negative in the search for functionality. Hence, this control voltage configuration would be completely ignored. By chance, the model 1L90N does give the AND functionality, however, with a large error in the current prediction. We observe a similar behaviour for XNOR, where all models up to 2L15N fail to predict functionality (not shown). Similarly, large errors in the prediction of the output current could result in false positives, generating control voltage configurations that solve the task in the model but fail the validation in the device. Both tasks are solved in model 2L30N with a separation smaller than 2 nA, however, these control voltage

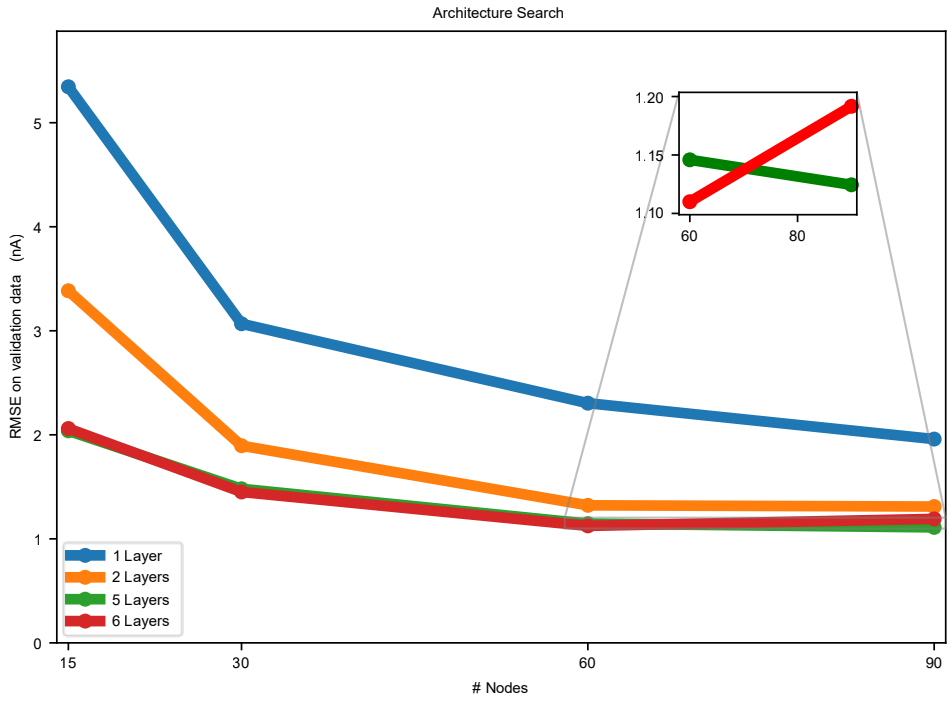


Figure 5.7: Performance of different DNN architectures. The root-mean-square error (RMSE) in the currents predicted by the trained DNNs as compared to the measured validation data. Errors are plotted versus the number of nodes per layer in the case of 1 to 6 layers. The inset shows the difference between DNNs with 5 and 6 hidden layers for 60 and 90 nodes per layer.

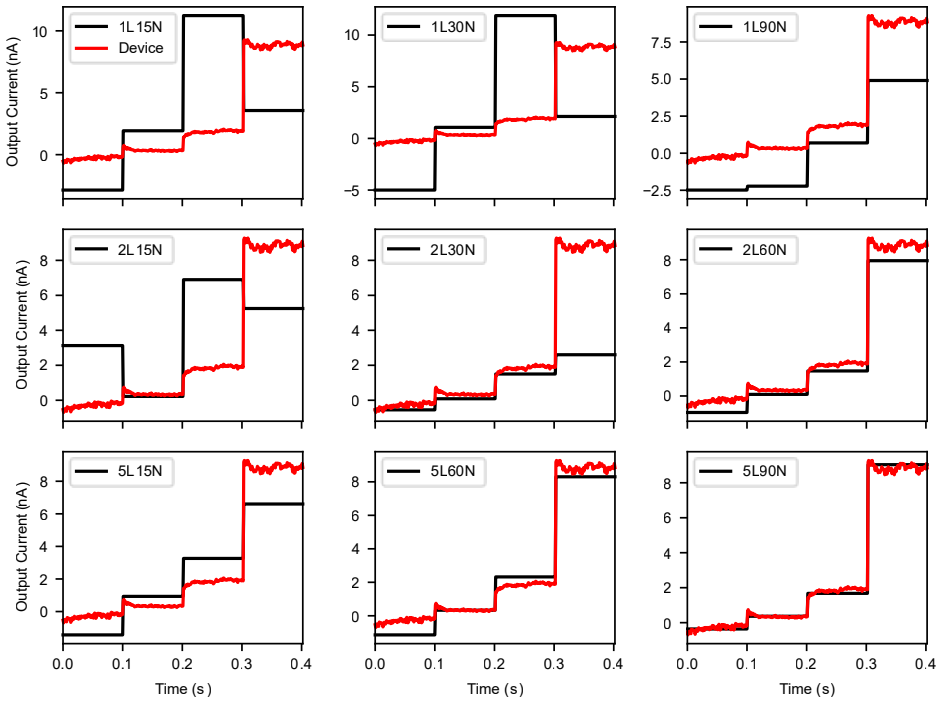


Figure 5.8: AND gate predictions of DNN models with different architectures. Black curves: predicted output from DNN models with architectures given in Figure 5.7 designated as xLyN, with x layers and y nodes. Red curve: measured output current (nA) of the device with the AND gate control voltages as in the main text (same for all panels).

configurations could be ignored by the searching procedure since the loss function optimizes for separation and false positive solutions with larger separations could be prioritised. All other tasks are solved with all models, so those voltage configurations happened to be robust to model errors due to the large differences in the "high" and "low" levels.

Naturally, if tasks require higher accuracy, the performance of the model becomes more relevant. Figure 5.9 shows the prediction of the output current given the same control voltages that solve the ring classification task as in the main text. We observe again a failure of the simpler models (up to 2L15N) to predict functionality with these control voltages. Coincidentally, the model 2L30N has low error for these inputs, predicting functionality with a relatively large gap. However, the models 2L60N and 5L15N fail to predict the correct separation of the classes, increasing the probability

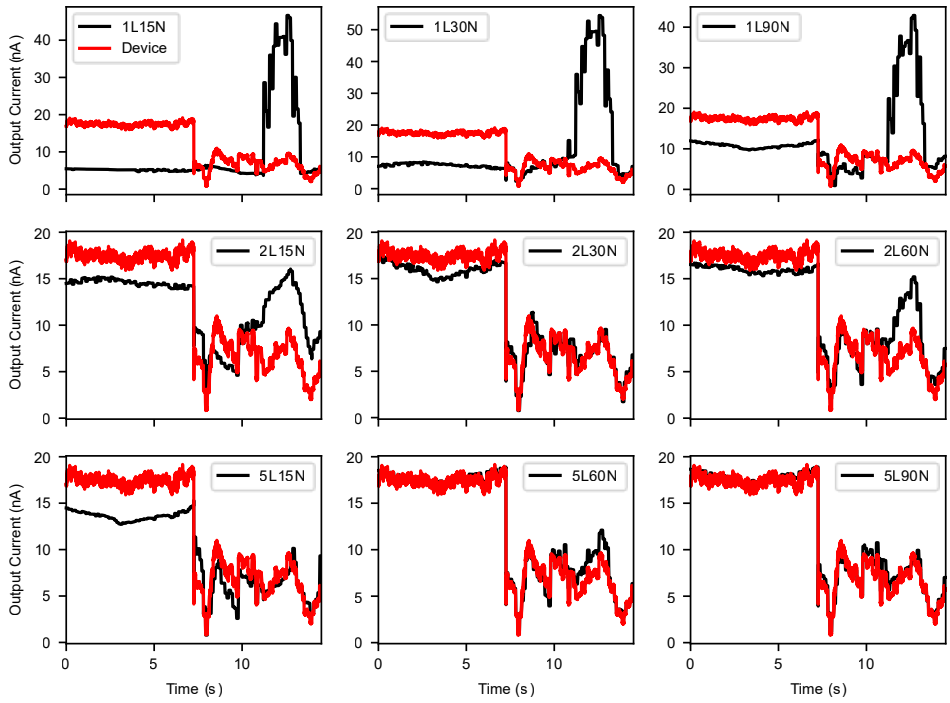


Figure 5.9: Model comparisons for the ring classification. Same representation as in Figure 5.8. For this task, more complex models are required to correctly predict the control voltage configuration.

that this solution would be ignored if false positives exist with larger separation between classes, i.e. only the models with the lowest RMSE correctly and reliably predict the functionality with large separability between classes.

The need for more complex models in tasks with even higher accuracy requirements is shown in Figure 5.10. Here, we compare the predicted outputs for the feature mapping task solved in the main text. The models used for the prediction of the output are those used to estimate the RMSE curves in Figure 5.7. We observe that in this example even mid-sized models like 2L30N and 5L15N fail (blue windows). There are two non-exclusive ways in which a model can fail in this task. First, if there is an incorrectly predicted separation of the features, the search will result in a false negative for this specific control voltage configuration. Second, even if the predicted features are well separated, the order of the features can be exchanged, for instance as

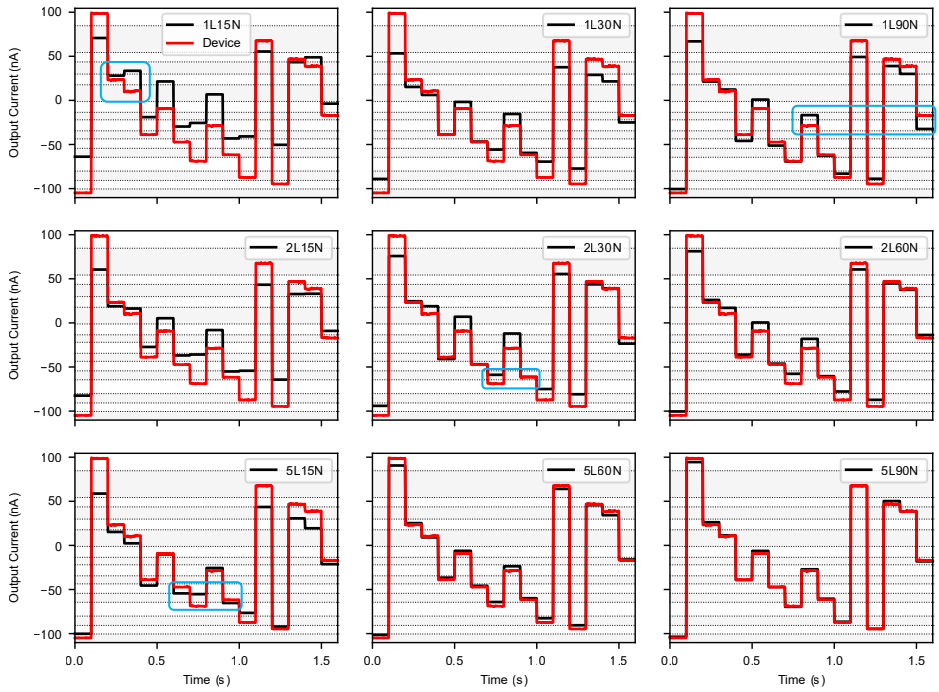


Figure 5.10: Model comparison for the 2×2 pixel feature mapping task. Same representation as in Figure 5.8, with the addition that the red curve represents the mean output over 10 measurements. Even the mid-sized models 2L30N and 5L15N fail in the functionality given the control voltage configuration obtained in the main text. The error in the models creates situations (examples of these are indicated by blue windows) where there is no meaningful separation predicted, which makes the voltage configuration a false negative, or there is an output current from the device that transposes the order of the predicted feature mapping, see e.g. 1L90N.

in model 1L90N. Although technically we would accept this solution because we only optimise for separability, the order of the features in the output would be permuted. Hence, the assignment of the predicted output after training a naive Bayes classifier would not correspond to the correct feature, making the functionality search rely more heavily on the naive Bayes classifier and the validation of the functionality obtained a posteriori.

Table 5.3: Control voltages for the alternative ring classification. Control voltages predicted by the DNN, applied to the device to verify the alternative ring classification in Figure 5.11.

Terminal	1	2	3	4	5
Control voltage (V)	0.10	-1.20	0.40	0.48	0.08

5.8.3 Alternative Ring Classification Functionality

Figure 5.5 in the main paper shows a ring classification functionality using V_4 and V_5 of the device as data inputs. To demonstrate that other electrode choices are also possible, Figure 5.11 shows an alternative ring classification functionality using V_6 and V_7 as inputs. The RMSE is in this case 0.83 nA. The control voltages for this alternative ring classification functionality are given in Table 5.3 and the scaling factor and offsets of the two input voltages in Table 5.4.

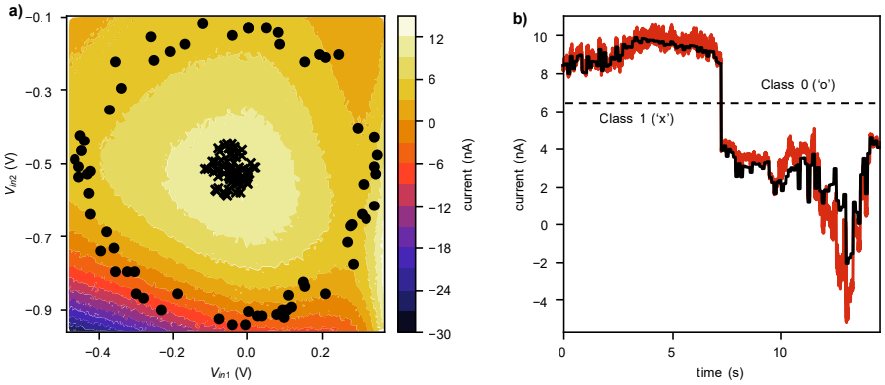


Figure 5.11: Alternative ring classification functionality. Ring functionality as in Figure 5.5 of the main text, but now with V_6 and V_7 from Figure 5.3b in the main paper as data input voltages $V_{in,1}$ and $V_{in,2}$. (a) 66 outer points (class '0', discs) and 66 inner points (class '1', crosses) to be classified, superimposed on a heat map of the device's output current. (b) Prediction of the current by the trained DNN (black curve) and verification measurement (red curve). The dash black line indicates a possible threshold value for the class assignment.

Table 5.4: Input scaling and offsets for the alternative ring classification. Scaling factor and offsets of the input voltages predicted by the DNN, applied to the device to verify the alternative ring classification in Figure 5.8.

Terminal	6	7
Scaling voltage (V)	0.42	0.42
Offset voltage (V)	-0.046	-0.517

5.8.4 Control Voltage Parameters of Optimised Functionalities in Main Text.

Table 5.5: Control voltages for Boolean logic. Control voltages (in V) predicted by the DNN, applied to the device to verify the Boolean logic gates in Figures. 5.3 and 5.4.

Terminal	1	4	5	6	7
AND	-1.19	0.15	-0.02	0.21	-0.36
NAND	-0.48	-1.06	0.23	-0.60	0.30
OR	-0.30	-1.20	-1.07	-0.27	-0.33
NOR	-1.14	-0.18	-0.05	0.30	0.28
XOR	-0.62	-0.17	-0.78	-0.63	0.26
XNOR	-1.20	0.11	-0.35	0.30	-0.27

Table 5.6: Control voltages for the ring classification. Control voltages predicted by the DNN, applied to the device to verify the ring classification in Figure 5.5.

Terminal	1	2	3	6	7
Control voltage (V)	0.28	0.60	-1.02	-0.02	0.69

Table 5.7: Affine transformation parameters for the ring classification. Scaling voltage and offsets of the input voltages predicted by the DNN, applied to the device to verify the ring classification in Figure 5.5.

Terminal	4	5
Scaling voltage (V)	-0.80	-0.80
Offset voltage (V)	-0.167	0.309

Table 5.8: Control voltages for the pixel feature mapping. Control voltages predicted by the DNN, applied to the device to verify the pixel feature mapping in Figure 5.6.

Terminal	1	6	7
Control voltage (V)	-1.20	0.10	-0.10

Table 5.9: Affine transformation parameters for the pixel feature mapping. Scaling voltages and offsets of the input voltages predicted by the DNN, applied to the device to verify the pixel feature mapping in Figure 5.6.

Terminal	2	3	4	5
Scaling voltage (V)	0.85	0.45	-0.07	0.41
Offset voltage (V)	-0.35	-0.85	0.46	-0.46

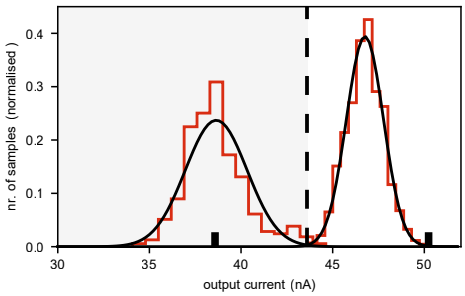


Figure 5.12: Decision boundaries for the pixel feature mapping. Zoom-in to the current region $30 - 52\ nA$ in the right panel of Figure 5.6b. The red lines show the current histograms of two-pixel features (0010 and 1000), with Gaussian fits shown by the black lines. The black dashed line shows the decision boundary current obtained with a naive Bayes classifier. The DNN predictions of the current for the two features are shown as thick black marks.

Table 5.10: Decision boundary currents for the pixel feature mapping. Currents defining the decision between different 2x2 pixel features in Figure 5.6b obtained with a naive Bayes classifier.

Pixel feature decision	Decision boundary current (nA)
1111 – 1011	-100.41
1011 – 0111	-90.74
0111 – 0011	-80.14
0011 – 1101	-64.51
1101 – 1001	-54.37
1001 – 0101	-43.59
0101 – 1110	-34.48
1110 – 0001	-22.06
0001 – 1010	-13.56
1010 – 0110	-1.19

References

[1] T. A. Baart, P. T. Eendebak, C. Reichl, W. Wegscheider, and L. M. K. Vandersypen. Computer-automated tuning of semiconductor double quantum dots into the single-electron regime. *Appl. Phys. Lett.*, 108, 2016. doi: 10.1063/1.4952624.

[2] S. S. Kalantre, J. P. Zwolak, S. Ragole, X. Wu, N. M. Zimmerman, M. D. Stewart, and J. M. Taylor. Machine learning techniques for state recognition and auto-tuning in quantum dots. *Npj Quantum Inf.*, 5, 2019. doi: 10.1038/s41534-018-0118-7.

[3] T. Botzem, M. D. Shulman, S. Foletti, S. P. Harvey, O. E. Dial, P. Bethke, P. Cerfontaine, R. P. G. McNeil, D. Mahalu, V. Umansky, A. Ludwig, A. Wieck, D. Schuh, D. Bougeard, A. Yacoby, and H. Bluhm. Tuning methods for semiconductor spin qubits. *Phys. Rev. Appl.*, 10, 2018. doi: 10.1103/PhysRevApplied.10.054026.

[4] C. J. van Diepen, P. T. Eendebak, B. T. Buijtendorp, U. Mukhopadhyay, T. Fujita, C. Reichl, W. Wegscheider, and L. M. K. Vandersypen. Automated tuning of inter-dot tunnel coupling in double quantum dots. *Appl. Phys. Lett.*, 113, 2018. doi: 10.1063/1.5031034.

[5] J. D. Teske, S. S. Humpohl, R. Otten, P. Bethke, P. Cerfontaine, J. Dedden, A. Ludwig, A. D. Wieck, and H. Bluhm. A machine learning approach for automated fine-tuning of semiconductor spin qubits. *Appl. Phys. Lett.*, 114, 2019. doi: 10.1063/1.5088412.

[6] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.C. Ruiz Euler, H. Broersma, P.r A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel. Classification with a disordered dopant-atom network in silicon. *Nature*, 577, 2020. doi: 10.1038/s41586-019-1901-0.

[7] S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nat. Nanotechnol.*, 10, 2015. doi: 10.1038/nnano.2015.207.

[8] O. Lykkebø, S. Nichele, and G. Tufte. An Investigation of Square Waves for Evolution in Carbon Nanotubes Material. volume ECAL 2015: the 13th European Conference on Artificial Life, pages 503–510, 07 2015. doi: 10.1162/978-0-262-33027-5-ch088.

[9] J. F. Miller, S. L. Harding, and G. Tufte. Evolution-in-materio: evolving computation in materials. *Evol. Intell.*, 7, 2014. doi: 10.1007/s12065-014-0106-6.

[10] S. Stepney. The neglected pillar of material computation. *Physica D*, 237, 2008. doi: 10.1016/j.physd.2008.01.028.

[11] J. P. Zwolak, T. McJunkin, S. S. Kalantre, J. P. Dodson, E. R. MacQuarrie, D. E. Savage, M. G. Lagally, S. N. Coppersmith, M. A. Eriksson, and J. M. Taylor. Autotuning of double-dot devices in situ with machine learning. *Phys. Rev. Applied*, 13, 2020. doi: 10.1103/PhysRevApplied.13.034075.

[12] D. T. Lennon, H. Moon, L. C. Camenzind, Liuqi Yu, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, E. A. Laird, and N. Ares. Efficiently measuring a quantum device using machine learning. *Npj Quantum Inf.*, 5, 2019. doi: 10.1038/s41534-019-0193-4.

[13] R. Durrer, B. Kratochwil, J.V. Koski, A.J. Landig, C. Reichl, W. Wegscheider, T. Ihn, and E. Greplova. Automated tuning of double quantum dots into specific charge states using neural networks. *Phys. Rev. Appl.*, 13, 2020. doi: 10.1103/PhysRevApplied.13.054019.

[14] M. Lapointe-Major, O. Germain, J. Camirand Lemyre, D. Lachance-Quirion, S. Rochette, F. Camirand Lemyre, and M. Pioro-Ladrière. Algorithm for automated tuning of a quantum dot into the single-electron regime. *Phys. Rev. B*, 102, 2020. doi: 10.1103/PhysRevB.102.085301.

[15] J. Darulová, S.J. Pauka, N. Wiebe, K.W. Chan, G.C Gardener, M.J. Manfra, M.C. Cassidy, and M. Troyer. Autonomous tuning and charge-state detection of gate-defined quantum dots. *Phys. Rev. Appl.*, 13, 2020. doi: 10.1103/PhysRevApplied.13.054005.

[16] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521, 2015. doi: 10.1038/nature14539.

[17] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2, 1989. doi: 10.1016/0893-6080(89)90020-8.

[18] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler. Big data of materials science: critical role of the descriptor. *Phys. Rev. Lett.*, 114, 2015. doi: 10.1103/PhysRevLett.114.105503.

[19] S. V. Kalinin, B. G. Sumpter, and R. K. Archibald. Big–deep–smart data in imaging for guiding materials design. *Nat. Mater.*, 14, 2015. doi: 10.1038/nmat4395.

[20] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh. Machine learning for molecular and materials science. *Nature*, 559, 2018. doi: 10.1038/s41586-018-0337-2.

[21] J. Carrasquilla and R. G. Melko. Machine learning phases of matter. *Nat. Phys.*, 13, 2017. doi: 10.1038/nphys4035.

[22] L.-F. Arsenault, A. Lopez-Bezanilla, O. A. Lilienfeld, and A. J. Millis. Machine learning for many-body physics: the case of the anderson impurity model. *Phys. Rev. B*, 90, 2014. doi: 10.1103/PhysRevB.90.155136.

[23] P. Werbos and P. John. Beyond regression: New tools for prediction and analysis in the behavioral sciences. phd dissertation, harvard univ.

[24] D. E. Rumelhart and J. L. McClelland. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362, 1987.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86, 1998. doi: 10.1109/5.726791.

[26] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron. Comput.*, EC-14, 1965. doi: 10.1109/PGEC.1965.264137.

[27] J. F Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, pages 167–176. IEEE, 2002. doi: 10.1109/EH.2002.1029882.

[28] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65, 1958. doi: 10.1037/h0042519.

[29] R. Li, L. Petit, D. P. Franke, J. Pablo Dehollain, J. Helsen, M. Steudtner, N. K. Thomas, Z. R. Yoscovits, K. J. Singh, S. Wehner, L. M. K. Vandersypen, J. S. Clarke, and M. Veldhorst. A crossbar network for silicon quantum dot qubits. *Sci. Adv.*, 4, 2018. doi: 10.1126/sciadv.aar3960.

[30] C. D. Hill, E. Peretz, S. J. Hile, M.G. House, M. Fuechsle, S. Rogge, M. Y. Simmons, and L. C. L. Hollenberg. A surface code quantum computer in silicon. *Sci. Adv.*, 1, 2015. doi: 10.1126/sciadv.1500707.

[31] M. Veldhorst, H. G. J. Eenink, C. H. Yang, and A. S. Dzurak. Silicon cmos architecture for a spin-based quantum computer. *Nat. Commun.*, 8, 2017. doi: 10.1038/s41467-017-01905-6.

[32] N. M. van Esbroeck, D. T. Lennon, H. Moon, V. Nguyen, F. Vigneau, L. C. Camenzind, L. Yu, D. M. Zumbühl, G. A. D. Briggs, D. Sejdinovic, and N. Ares. Quantum device fine-tuning using unsupervised embedding learning. *New Journal of Physics*, 22, 09 2020. doi: 10.1088/1367-2630/abb64c.

[33] H. Moon, D. T. Lennon, J. Kirkpatrick, N. M. van Esbroeck, L. C. Camenzind, Liuqi Yu, F. Vigneau, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, D. Sejdinovic, E. A. Laird, and N. Ares. Machine learning enables completely automatic tuning of a quantum device faster than human experts. *Nat. Commun.*, 11, 2020. doi: 10.1038/s41467-020-17835-9.

[34] J. Darulova, M. Troyer, and M. C. Cassidy. Evaluation of synthetic and experimental training data in supervised machine learning applied to charge state detection of quantum dots. 2020. doi: 10.48550/ARXIV.2005.08131.

[35] O. Tsilipakos, A. C. Tasolamprou, A. Pitilakis, F. Liu, X. Wang, M. S. Mirmoosa, D. C. Tzarouchis, S. Abadal, H. Taghvaee, C. Liaskos, A. Tsioliaridou, J. Georgiou, A. Cabellos-Aparicio, E. Alarcón, S. Ioannidis, A. Pitsillides, I. F. Akyildiz, N. V. Kantartzis, E. N. Economou, C. M. Soukoulis, M. Kafesaki, and S. Tretyakov. Toward intelligent metasurfaces: the progress from globally tunable metasurfaces to software-defined metasurfaces with an embedded network of controllers. *Adv. Opt. Mater.*, 8, 2020. doi: 10.1002/adom.202000783.

[36] Ni-daqmx python documentation (national instruments corp., 2017), . URL `https://nidaqmx-python.readthedocs.io/en/latest`.

[37] Skynet library (darwin team of the nanoelectronics group, univ. of twente, 2020), . URL `https://github.com/BraiNEdarwin/SkyNEt`.

[38] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. 2020. doi: 10.48550/ARXIV.1710.09282.

[39] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014. doi: 10.48550/ARXIV.1412.6980.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. 12 2019. doi: 10.48550/arXiv.1912.01703.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

6

Dopant Network Processing Units: Towards Efficient Neural-network Emulators with High-capacity Nanoelectronic Nodes

The rapidly growing computational demands of deep neural networks require novel hardware designs. Recently, tuneable nanoelectronic devices were developed based on hopping electrons through a network of dopant atoms in silicon. These "Dopant Network Processing Units" (DNPUs) are highly energy-efficient and have potentially very high throughput. By adapting the control voltages applied to its electrodes, a single DNPU can solve a variety of linearly non-separable classification problems. However, using a single device has limitations due to the implicit single-node architecture. This paper presents a promising novel approach to neural information processing by introducing DNPUs as high-capacity neurons and moving from a single to a multi-neuron framework. By implementing and testing a small multi-DNPU classifier in hardware, we show that feed-forward DNPU networks improve the performance of a single DNPU from 77% to 94% test accuracy on a binary classification task with concentric classes on a plane. Furthermore, motivated by the integration of DNPUs with memristor crossbar arrays, we study the potential of using DNPUs in combination with linear layers. We show by simulation that an MNIST classifier with only 10 DNPU nodes achieves over 96% test accuracy. Our results pave the road towards hardware neural-network emulators that offer atomic-scale information processing with low latency and energy consumption.

This Chapter is published as: H.-C. Ruiz Euler, U. Allegre Ibarra, B. van de Ven et al. Dopant Network Processing Units: Towards Efficient Neural-network Emulators with High-capacity Nanoelectronic Nodes. Neuromorph. Comput. Eng. **1**, 024002. doi: 10.1088/2634-4386/ac1a7f

Contributions: Device fabrication, initial device characterisation and measurement support.

6.1 Introduction

The success of deep neural networks (DNNs) comes with an exponential increase in the number of parameters and operations, which brings along high energy costs, high latency, and massive hardware infrastructure. Moreover, due to a slowdown of Moore's law, the gap between the computational demands of DNNs and the efficiency of the hardware used to implement them is expected to grow. There is a broad spectrum of research on hardware acceleration focused on obtaining state-of-the-art performance in DNNs, while reducing associated costs. Solutions range from traditional approaches, which can be implemented on the short-term, to novel long-term approaches trying to address fundamental problems such as that of the Von Neumann bottleneck or the slowdown of Moore's law [1].

6.1.1 State-of-the-Art Approaches.

General-purpose hardware approaches for DNN acceleration typically employ a variety of temporal architectures to improve parallelism of multiply-accumulate (MAC) operations involved in convolutions and fully connected layers [2]. Specialised hardware approaches improve on the bottlenecks from the design of general computing. Since the energy consumption is dominated by the data movement during computation, these approaches are mostly focused on spatial architectures, reducing energy consumption by increasing the data reuse from low-cost memory hierarchy via optimized data-flows [3]. Specialised hardware encompasses FPGA-based acceleration [4], ASIC-based acceleration [5, 6, 7, 8, 9, 10], or a combination of both [11]. Furthermore, the development of specialised hardware enables DNN algorithm optimisation techniques to be jointly designed with the hardware [12, 13, 14, 15].

6.1.2 Neuromorphic Computing.

To reduce the impact of the data-movement bottleneck, some research aims at bringing memory closer to the computation, or even integrating the memory and the computation into a single architecture [3]. The latter approach encompasses the use of memristors, non-volatile electronic memory devices that can integrate MAC operations into the memory [16, 17, 18]. Recent developments show the potential of memristor crossbar arrays for implementing DNN connectivity fully in hardware [19]. Other research efforts try to overcome the hardware efficiency problem by taking a variety of fundamentally different approaches, ranging from spiking neural networks, which use sparse binary signals to compute asynchronously and in a massively parallel manner, to new unconventional methods for material-based computation [20, 21, 22, 23]. From all these approaches, spiking neural networks are the most mature but, although they show high energy efficiency and low latency [24], they tend not to support state-of-the-art DNN models, and their efficient training and proper benchmarking tends to be problematic [3, 25].

6.1.3 A Promising Novel Technology.

Recently, tuneable nanoelectronic devices were developed capable of classifying linearly non-separable data, e.g. XOR [26, 27]. These so-called dopant network devices are projected to have an energy efficiency in the order of 100 TOPS/W and a bandwidth of over 100 MHz, making them an attractive candidate for novel, unconventional hardware solutions for information processing. Interestingly, the ability of these designless devices to perform the XOR operation coincides with that of individual human neocortical neurons, a recent observation that contradicts the conventional belief that this computation requires multi-layered neural networks [28]. Motivated by the above observations, we envision the usage of dopant network devices as *high-capacity* nodes in a hardware architecture that emulates neural networks. The implementation of these *neural-network emulators* would have several advantages. First, the expected small footprint, high throughput and energy efficiency would allow portability and low latency. Second, massive parallelisation could be possible using many independent devices. Third, computation is performed in-materio, i.e. by physical processes in the devices. Thus, the need of explicit arithmetic operations is greatly reduced, in particular if this technology is combined with memristor crossbar arrays. Moreover, in-materio computations could bypass the need of data management at the inference step because the learned parameters would be a fixed, physical characteristic of the system and computation would be reduced to physical processes transforming and propagating information. Finally, high-capacity nodes may allow more compact neural network architectures that could bring additional benefits in terms of performance and efficiency.

6.1.4 Towards Novel Neural-Network Emulators.

In this paper, we take the first steps towards realising neural-network emulators with dopant network devices, giving new insights in its potential for this purpose. Section 6.2 reviews the state-of-the-art of this nanotechnology, which we will henceforth call *dopant network processing units* (DNPUs). In Section 6.3, we estimate the capacity of these complex, highly non-linear computational units in terms of an empirical estimate of the Vapnik–Chervonenkis dimension for binary classification of two-dimensional data. This general measure can be used to benchmark the computational capabilities of small material-based systems. In order to study the viability of DNPU interconnection, Section 6.4 demonstrates how DNPU feed-forward network architectures can be designed, trained and implemented in hardware to solve classification tasks more accurately than a single device. In Section 6.5, we explore novel, compact architectures that would reduce the number of parameters and/or operations, if inference were implemented with DNPU devices. We show, by simulation, that these high-capacity nodes allow for a classifier of hand-written digits from the MNIST dataset with high accuracy, using only 10 nodes. Section 6.6 discusses potential extensions of this work to large-scale neural-network emulators and their benefits.

6.2 Dopant Network Processing Unit

The basis of a DNPU is a lightly doped (n- or p-type) semiconductor with a nano-scale active region contacted by several electrodes, see Figure 6.1. Different materials can be used as dopant or host and the number of electrodes can vary. In this paper, we use a boron-doped silicon (Si:B) DNPU with an active region of 300 nm in diameter and the electrode configuration represented in Figure 6.1. Once we choose a readout electrode, the device can be activated by applying voltages to the remaining electrodes, which we call activation electrodes. The dopants in the active region form an atomic-scale network through which the electrons can hop from one electrode to another. This physical process results in an output current at the readout that depends non-linearly on the voltages applied at the activation electrodes. By tuning the voltages applied to some of the electrodes, the output current can be controlled as a function of the voltages at the remaining electrodes. This tunability can be exploited to solve various linearly non-separable classification tasks [26, 29].

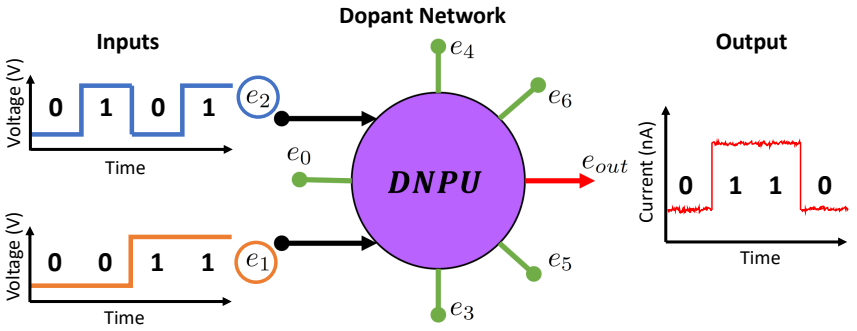


Figure 6.1: Sketch of a DNPU with eight electrodes [26], where e_{out} is the *readout* electrode and the others can be either *input* or *control* electrodes, e.g. $e_{1,2}$ and $e_{0,3-6}$, respectively. To implement a classifier, voltages are applied to the input electrodes representing the features of the data, e.g. 0 and 1. Applying a learned voltage configuration to the control electrodes implements the classifier, e.g. XOR, as an output current representing the classes 0 and 1.

6.2.1 Nonlinear Classification with DNPUs.

Let us assume that we want to create an XOR classifier using a single DNPU [26]. For concreteness, let us consider a DNPU with eight electrodes, which are divided into seven activation electrodes e_{0-6} and a single readout electrode e_{out}, as shown in Figure 6.1. From the activation electrodes, e_1 and e_2 are chosen as *data input electrodes*. These receive voltage-encoded signals representing the binary input features of the XOR classification task. We call these signals the *input voltages*. The remaining

activation electrodes ($e_{0,3-6}$) are selected as *control electrodes*, and are used to tune the relation between the input voltages and the output current. The voltage values applied at these electrodes are the learnable parameters and we call them *control voltages*. Historically, DNPUs have been trained exploiting the concept of evolution-in-materio [30], which adopts a genetic algorithm to find adequate control voltages directly on-chip [26, 27]. A more recent approach [29] creates DNN surrogate models of DNPUs to predict the output current from the voltages applied to the activation electrodes. These surrogate models enable learning the control voltages *off-chip* by gradient descent (GD) using standard deep-learning packages. We use PyTorch [31] for both, training the DNN surrogate model and the off-chip training in all experiments. We can enable functionality, i.e. inference, by applying the control voltage values found during off-chip GD training, along with the corresponding data input voltages, Fig. 6.1. As shown in [26], the functionality of the DNPU remains consistent over time, even after switching to other voltage configurations or turning off the device. This means that control voltage values for a particular task only need to be found once.

6.2.2 DNPU Surrogate Models.

Before we can use the off-chip training method to find functionality in DNPUs, we must create a DNN surrogate model of the device [29]. The input and output nodes of the model correspond to the DNPU's activation and output electrodes, respectively (Fig. 6.2). This model will map the input voltages to the output current, so the behaviour of the physical device is reproduced. With this "digital copy" we can then train the DNPU off-chip, i.e. without using the device during training. For modelling, we sampled 4.5 million input-output pairs in a voltage range of $[-1.2, 0.6]$ V for electrodes e_{0-4} and $[-0.7, 0.3]$ V for $e_{5,6}$. These voltage ranges are determined by the electrical properties of the device. With these data, we trained a feed-forward DNN–five hidden layers, each having 90 ReLU nodes–for 500 epochs in batches of 128 and a learning rate $\eta = 0.0005$. Using an independently sampled test set of 4.5×10^5 samples, the root-mean-squared error is found to be 1.4 nA, corresponding to 0.35% of the total current output range between -300 and 100 nA. This prediction error comes from measurement contributions and physical noise in the output current.

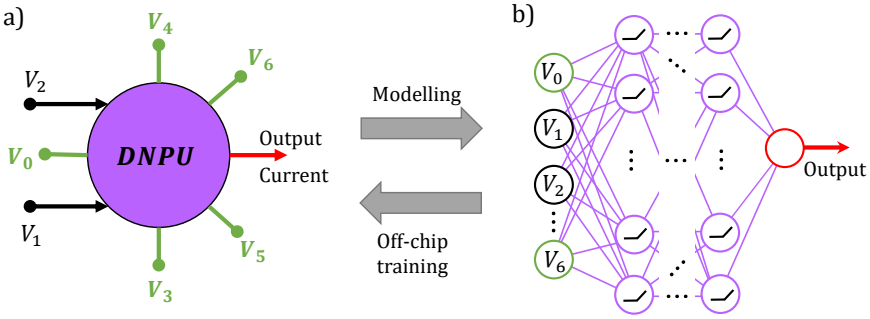


Figure 6.2: DNPU modelling for off-chip training. a) Cartoon of the DNPU with control and input electrodes represented as green and black pins. The electrodes have the same configuration as in Fig. 6.1. b) Surrogate model of the DNPU. The activation electrodes of the DNPU (a) are assigned to inputs of the DNN model. Given data, we can train this DNN to map the voltages at the activation electrodes to the output current of the device. At this stage, all inputs of the DNPU are treated equally, i.e. there is no assignment yet of control electrodes. To train the DNPU, we assign certain inputs of the DNN model to learnable parameters (green), which are found via off-chip training targeting the desired functionality, e.g. classification. The values found represent control voltages, which are then applied to the corresponding control electrodes in a). Both training the model and the off-chip training are done using conventional deep learning techniques.

6.2.3 Training DNPUs Off-Chip.

The off-chip GD training technique works as follows [29]. The internal weights and biases of the surrogate model, shown in Figure 6.2b with purple colour, are kept frozen. We attach learnable *control parameters* to the model's inputs corresponding to control electrodes–green pins and nodes in Figure 6.2. Learning the control parameters via GD exploits the fact that the gradient can be back-propagated to the input nodes with control parameters attached to them. For all the results presented in this paper, we used the following procedure and hyperparameters, unless explicitly mentioned otherwise. We regularized during off-chip training the control parameters with an L1-penalty outside of the working voltage ranges of the device. The penalty strength is set to $\alpha = 1.0$. We use Adam [32] with the default parameters. After off-chip training, the control parameters found are validated by implementing the inference step on the device. This involves measuring the complete data set by arranging the samples sequentially in time, as shown in Figure 6.1. Then, the voltages representing each sample are applied one after another, while the control voltages remain fixed throughout the whole validation step. Each data point is measured by applying its corresponding input voltages to the input electrodes for 0.8 seconds with a sampling frequency of 100 Hz. Currently, the experimental setup has a bandwidth of around

40 Hz, however, we decided to have longer readout times to account for the noise in our accuracy estimates. There are already considerations to increase energy efficiency and bandwidth (theoretical limit: 100 MHz, [26]).

6.3 Capacity of a Single Dopant Network Processing Unit

In this section, we show that a single DNPU has a computational capacity comparable to that of a small artificial neural network. For this, we estimate the capacity of the DNPU surrogate model, the physical device and two small neural networks. As a measure of the computational capacity, we use an empirical estimate of the Vapnik-Chervonenkis (VC) dimension. More specifically, given N points, they can be mapped to $\{0,1\}$ in 2^N ways, i.e. there are 2^N possible classifiers. Then, the capacity of a computational system for a given N is defined as the fraction C_N of classifiers that can be realized by the system. Loosely speaking, the VC-dimension is defined as the largest N such that $C_N = 1$ [33]. By searching for all the classifiers, we can give an empirical estimate of the computational system's capacity in terms of the VC-dimension. We search for all classifiers on a fixed number N of data points in the 2-dimensional plane, where $N = 4, \ldots, 10$, see Figure 6.3a. These points are chosen such that they fall within the working voltage range of the input electrodes of the DNPU (e_1 and e_2). The remaining electrodes $e_{0,3-6}$ are used as control electrodes. Since XOR is the first non-trivial case, we start with $N = 4$ and select the first four points of the list given in Figure 6.3a. For $N = 5$, we add the next point on the list and so on. Given N, we train each classifier using binary cross-entropy loss. This requires passing the output of the DNPU surrogate model through a *decision node* consisting of a batch norm layer with a learnable affine transformation and a logistic function. We train for 1,500 epochs with full batch, a learning rate $\eta = 0.03$ and the Adam moment parameters $(\beta_1, \beta_2) = (0.995, 0.999)$. The number of epochs is chosen to give all 2^N classifiers enough time to converge. However, no systematic hyper-parameter search was performed. Furthermore, we allow for 15 attempts to find each classifier, which is considered to be found if its accuracy is 100%.

Estimating the device's capacity requires special attention because of the noise in the current output. The off-chip training is an extension of [29] where we account for the noise in the output current. While training, we add Gaussian noise to the model's output and reduce it after every attempt by a factor depending on the attempt's number n: $\sigma^2_{n+1} = (1 - \frac{n+1}{15})\sigma^2_n$ with σ^2_{n+1} the noise variance at attempt $n+1$ and $\sigma^2_0 = 1$. This forces the training procedure to find robust solutions with a sufficient signal-to-noise ratio. The solutions obtained are validated on the physical device to determine if the classifier is found. In this case, a classifier is considered to be found if its accuracy is above a threshold $\theta = 1 - 0.5/N$, i.e. if more than 50% of the measurements corresponding to one of the N points can be classified correctly. before, the predicted label is determined by the decision node which we re-train with the measured data, leaving out at random 20% of the measurements as a test set

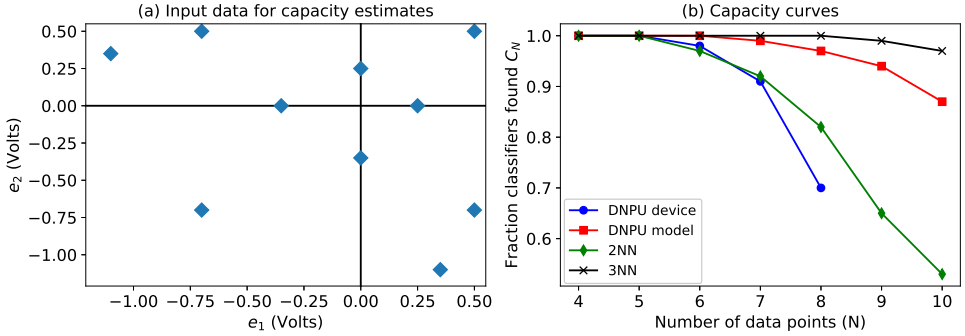


Figure 6.3: a) Input data for capacity estimates: {(-0.7,-0.7), (-0.7,0.5), (0.5,-0.7), (0.5,0.5), (-0.35,0), (0.25,0), (0,-0.35), (0,0.25), (-1.1,0.35), (0.35,-1.1)}. The classifiers should map these points to {0,1} in all possible ways. b) Capacity of the DNPU surrogate model, the physical DNPU, and neural networks with a hidden layer of two (2NN) and three (3NN) neurons.

to estimate the final accuracy on hardware. Once all classifiers are validated, we perform a new search for any failed cases and repeat this procedure one more time. The capacity of the DNPU is compared to that of two small neural networks with one hidden layer of 2 and 3 neurons. Both networks have logistic activation functions and are trained with the same procedure as the DNPU's surrogate model.

Figure 6.3b shows the capacity estimates for all four systems. The capacity of the physical device follows closely that of the neural network with two hidden nodes, both having a VC dimension of at least 5. The DNPU surrogate model has a VC dimension of at least 6 and the network with three hidden neurons has a VC dimension of at least 8. As expected, we observe a steeper decay in the capacity of the device due to noise. However, the DNPU surrogate model has a much larger capacity than a network with two hidden neurons. This is consistent with the observation in [29] that the nano-electronic device can solve binary classification problems with closed decision boundaries, which requires conventionally at least three hidden neurons. Hence, we can consider the DNPU capacity curves shown in Figure 6.3b as upper and lower empirical limits of their capacity. We want to remark at this point an important benefit of using the DNPU device for inference. Classification with the DNPU in this experiment requires the evaluation of the device, and only two arithmetic operations at the decision node, but no explicit operation is required to compute the internal representation of the data that allows for linear separability. Hence, we count three operations to implement each classifier. In contrast, the neural network with two hidden nodes requires at least 12 arithmetic operations without counting the contribution of non-linear activation functions. The difference is even more dramatic when compared to a network with three hidden nodes, which requires 18 operations. Inter-

estingly, we also observe that we can solve classification tasks with fewer parameters, see first three columns in Table 6.1.

Table 6.1: Computational requirements for different architectures as the number of (arithmetic) *operations* and *parameters* required in their implementation. For all DNPU classifiers, we counted the evaluation of one DNPU as an operation. On the contrary, we disregarded the non-linearities in the neural network architectures 2NN, 3NN and ANN-7D when counting the operations. The networks 2NN and 3NN are small networks with two and three hidden neurons respectively, see Fig. 6.3b. The networks ANN-7D and DNPU-7D are MNIST classifiers described in Section 6.5.

Architecture	DNPU	2NN	3NN	DNPU-7D	ANN-7D
nr. operations	3	12	18	10	140
nr. parameters	7	9	13	0	80

6.4 Multi-DNPU Network

This section explores the potential of DNPU scalability by comparing the performance of a single DNPU device with a feed-forward network architecture of five DNPU devices. Particularly, we consider a linearly non-separable binary classification problem consisting of two classes that fall into two concentric circles with a given separation *gap*, see Figure 6.5a. In Ref. [29], it was shown how a single DNPU is sufficient to resolve this classification problem with 100% accuracy for a 400 mV gap. For the purpose of this section, the gap has been significantly reduced from 400 mV to 6.25 mV, requiring a more accurate decision boundary in order to be solved. The data used consist of around 400 input points and their corresponding binary labels, equally divided into training and test sets. Each class in each set has around 100 i.i.d. samples generated from random draws over a uniform distribution on their respective concentric circular areas, as shown in Figure 6.5a. The same data have been used for both classifiers, the single DNPU and the DNPU network. All nodes in both classifiers have the same input, control and output electrode configurations, namely e_1 and e_2 for the input, $e_{0,3-6}$ for control and e_{out} for the output, see Figure 6.1.

The DNPU network consists of two input nodes, two hidden nodes and one output node, see Figure 6.4. This 2-2-1 architecture is implemented on hardware by time-multiplexing the single DNPU to evaluate each node. That is, we mimic the 2-2-1 architecture using the same device, but changing sequentially the control voltages corresponding to each node. Each input unit receives a copy of the input data. Then, each unit in the hidden layer receives the outputs of the input layer. The output of each unit in the input and hidden layers is standardized and mapped linearly to the input voltage range of the units in the next layer. The last node receives the outputs

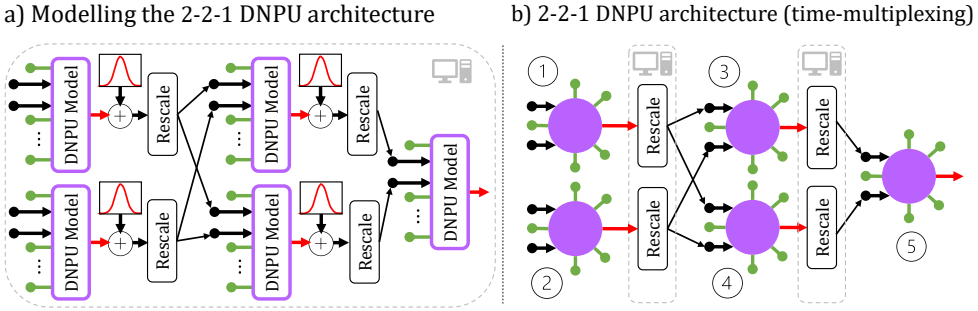


Figure 6.4: a) Modelling the 2-2-1 DNPU architecture for off-chip training. Each node is composed of the same DNPU surrogate model and postprocessing steps. The latter adds to the current prediction a random number drawn from a Gaussian distribution with standard deviation given by the prediction error. This approximation of the current is then transformed to a voltage value by rescaling and offsetting. b) The corresponding 2-2-1 DNPU architecture is implemented "virtually" by time-multiplexing a single physical device. Each node is composed of the same physical DNPU and a rescaling and offsetting made by a computer. The rescaling and offsetting are the same as those in the off-chip training model. The numbers next to each node represent the sequence in which each node is operated to implement the virtual architecture. The black arrows in a) and b) indicate how the information flows.

of the two hidden units. This network has a total of 25 control parameters, distributed through the five electrodes $e_{0,2-6}$ in each node. To demonstrate the capabilities of the DNPU explicitly, we tackle training in two steps. First, we train the classifiers to separate the classes as much as possible using the negative of the Fisher's linear discriminant criterion as the loss function. This shows directly the capability of both DNPU systems to separate linearly non-separable classes. Second, once the data are separated in the output of the DNPU systems, the decision threshold for assigning the class is relegated to a simple logistic neuron.The first training step uses 400 epochs with full batch updates and a learning rate of $\eta = 0.0065$. There was no systematic hyper-parameter optimization. To account for the noise in the physical device, the classifiers are trained with Gaussian noise added to the output of each node and with a variance equal to the mean-squared test error of the DNPU model, namely $\sigma^2 = 1.97$.Finally, the logistic neuron is trained on the standardized output of the DNPU system and tested to estimate the accuracy of the classifier.

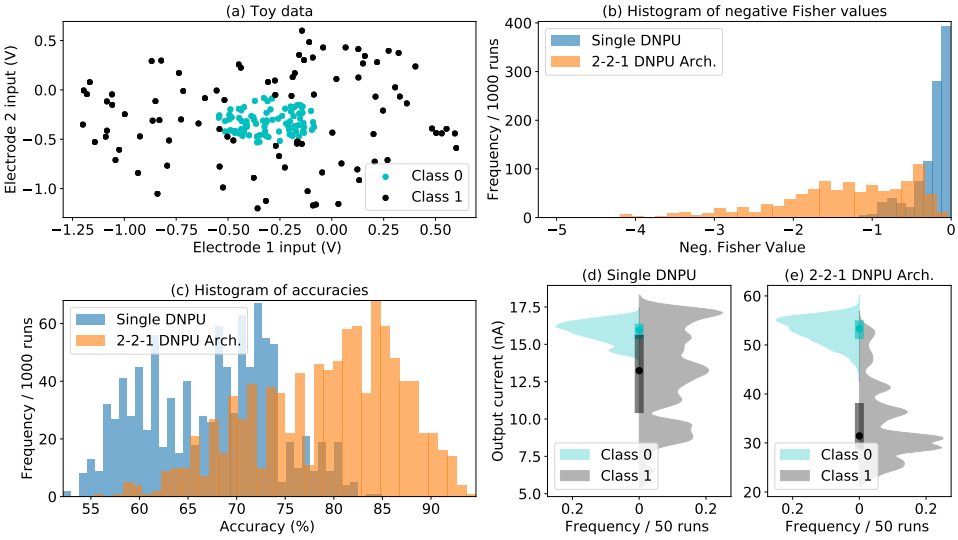


Figure 6.5: a) Input data used in the experiment with a gap of 6.25 mV between the two classes. b) Histogram of negative Fisher values. c) Histogram of accuracies. Both b) and c) are results from 1000 trials of the off-chip training, i.e. both the single DNPU and the 2-2-1 DNPU architecture are modelled. d) Outputs measured over 50 runs on a hardware DNPU using the control voltages obtained from the best result after 1,000 trials off-chip training. The outputs are separated by classes on each side of the violin plot. e) Outputs measured by time-multiplexing the hardware DNPU to implement the 2-2-1 DNPU architecture. The data are obtained with the same procedure as in d).

For each classifier, we run 1,000 off-chip training trials to compare the distribution of their solutions. Figure 6.5b shows the histogram of the negative Fisher values over all trials for both classifiers. The best performance obtained for a single device is -1.16, while the 2-2-1 architecture obtains -5.23, which is around 4.5 times lower, meaning a better separation between classes. Figure 6.5c shows the accuracy histograms over all trials for both classifiers. The highest accuracy in 1,000 trials of the off-chip training is for a single DNPU 85%, while the best 2-2-1 architecture has an accuracy of 95%, ten percent points more. Both results in Figures 6.5b,c show a significant improvement of the classification performance when using the 2-2-1 DNPU architecture in comparison to using a single DNPU. The best result of each classifier (minimum Fisher value and maximum accuracy over the 1,000 trials) is selected for validation on hardware. Validation is performed for both training and test sets. The measurements for the 200 samples in each dataset give approximately 16,000 current values. Similar to the two-step procedure in the off-chip training, the device's output separates the input voltages of the training set into two distinct current levels. Then, to fine-tune the classifier on hardware, we re-train the logistic output neuron on the standardized

current outputs. To evaluate the performance on the test set, 50 validation trials are measured over the entire set for each classifier. All these measurements are then passed through the logistic neuron of the classifier to obtain the predicted labels. The validation of the single DNPU classifier on hardware yields an accuracy of 77% on both the training and test sets. Figure 6.5d shows the device's output distribution per class over the 50 validation trials using the test set. Since class 0 has a broad distribution that overlaps entirely with that of class 1, we observe a failure to separate the classes properly for a 6.25 mV gap. The validation of the 2-2-1 DNPU architecture gives an accuracy of 97% on the training set and an accuracy of 94% on the test set. Figure 6.5e shows the output current distribution per class over the 50 trials. Contrary to the observations in Figure 6.5d, the 2-2-1 DNPU architecture gives a good separation of the classes, i.e. their dominant modes in the output current distribution are well separated and have only some overlap at currents around 45 nA. The validation of both classifiers shows consistent accuracy results. We estimated the variance of the classification accuracy on hardware over the 50 validation trials. We observe typical fluctuations in the accuracy of one percent point for both, the single DNPU and the 2-2-1 DNPU architecture. These fluctuations are expected due to the noise in the output current.

6.5 DNPU Classifier for MNIST

In this section, we explore the potential of high-capacity neurons in combination with linear layers to process high-dimensional data efficiently. This approach is motivated by a possible integration of DNPUs with technologies that can efficiently implement multiply-accumulate operations, such as memristor crossbar arrays [19], optical [34] and flash memory approaches [35]. DNPUs could act as tuneable, complex activation functions that complement other existing technologies to implement fully material-based neural networks with computationally efficient high-capacity neurons. We show by simulation in PyTorch how a DNPU network with only 10 high-capacity neurons can be used for classification on the MNIST data set, consisting of hand-written single digit (0-9) images of size $28 \times 28 = 784$ pixels in grey scale. There are 60,000 and 10,000 images in the training and test data, respectively. The former was split into 55,000 and 5,000 samples for training and validation, respectively. Besides flattening the images, we standardize the grey scale of the pixels to values between -0.5 and 0.5. No other pre-processing is implemented. Our DNPU classifier consists of a linear layer of size 784x30 and no bias parameters. This layer linearly combines all pixels in the image. The resulting *linear features* are fed into an output layer consisting of 10 DNPU nodes, each representing one class $\{0, \ldots, 9\}$ and receiving three linear features, see Figure 6.6a. The input assignments to the DNPU models are the same for all nodes in the classifier, namely, the inputs corresponding to V_0, V_3, V_4 in Figure 6.2b are assigned as data inputs and control parameters are attached to all other inputs. The DNPU classifier is trained using cross-entropy loss. We include weight decay for the linear layer parameters with a regularisation factor of 0.1 to force its

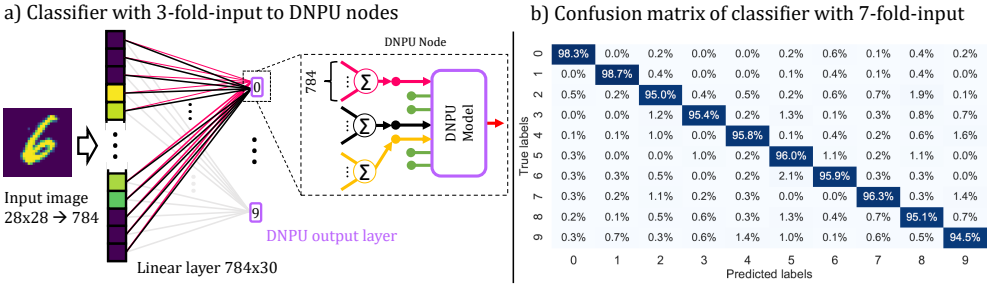


Figure 6.6: DNPU classifier consisting of a linear input layer and a DNPU output layer with 10 DNPU nodes; a) Classifier with 3-fold-input to each DNPU node. The 784 pixels of the images are linearly combined into 30 linear features in the linear layer. Each of the 10 DNPU nodes receives three linear inputs, i.e. out of the 30 linear features, the first three go to the first DNPU (pink, black and yellow in the dashed-line box), the second three go to the second DNPU and so on. For clarity not all connections of the linear layer are shown, but we remark that all these linear features are independent; b) Confusion matrix for MNIST classification of the DNPU classifier with 7-fold-input (DNPU-7D). As in a) but now the linear layer has a size of 784x70, and each DNPU node receives seven linear features instead of three.

outputs to fall within the working voltage ranges of DNPUs, which are typically between -1 and $+1$ Volts. We trained for 80 epochs with a learning rate $\eta = 2\times10^{-5}$ and a mini-batch of 64 images. The accuracy after training is 97% (95%) on the training (validation) set and 95% on the test set. There was no systematic hyper-parameter optimization, but we observed that this result is fairly consistent across different values of the learning rate and mini-batch size. Nevertheless, a learning rate that is two orders of magnitude larger decreases performance significantly. As a benchmark, we compare our DNPU classifier to a neural network with a similar architecture consisting of a hidden layer with 30 ReLU-neurons and an output layer consisting of 10 linear neurons, each connecting locally to three neurons of the hidden layer. Hence, instead of the DNPU model in Fig. 6.6, we have three ReLU-neurons each receiving a linear feature (pink, black, yellow). Their activation is then linearly combined into the activation of the corresponding linear output neuron. We denote this architecture ANN-3D. In this case, we have bias parameters in the hidden and the output layers. We train ANN-3D similar to the DNPU classifier with the difference of a higher learning rate $\eta = 3\times10^{-4}$. This neural network achieves 95% on the test data and 96% (95%) on the training (validation) set with a similar number of parameters to that of the DNPU classifier. Hence, in this case, the control parameters in the DNPU do not provide an increase in capacity. We attribute this to the reduced number of features available to each output node. The linear dimensionality reduction mixes the data such that the resulting class distributions are overlapped, making classification difficult.

Increasing the number of linear features fed to the DNPU nodes from three to seven (V_{0-6} in Figure 6.2b) increases accuracy to 96% on the test set and 98% (96%) on training (validation) set. We denote this DNPU classifier with 7-fold-input by DNPU-7D. Figure 6.6b shows the confusion matrix for DNPU-7D. We observe that the typical "confusions" are also present here: the digit 9 is misclassified 2.4% of the cases as the digits 4 and 5, and the digit 7 is misclassified 2.5% of the cases as the digits 2 and 9. For comparison, we use a similar architecture to ANN-3D, but we increase the number of linear features per output node to seven. Thus, this network has a hidden layer with 70 ReLU-neurons feeding to 10 linear nodes in groups of seven. This architecture, denoted by ANN-7D, has 96% accuracy on the test set and 98% (96%) on training (validation) set. Hence, as before, both the DNPU-7D and the ANN-7D have similar performance. Nevertheless, neglecting the non-linearity functions, the ANN-7D network must perform 14 arithmetic operations (seven multiplications and seven additions) per output node to compute its activation, while a physical realisation of DNPU-7D would require a single evaluation per node. Hence, taking the evaluation of the DNPU as a single operation, the DNPU-7D needs 10 operations for the output layer, while the ANN-7D requires 140 operations. In addition, each output node in the ANN-7D network requires 8 parameters (seven synaptic weights plus one bias), giving a total of 80 parameters in the output layer, while the DNPU device with a 7-fold-input would not require any additional parameters, see Table 6.1. Notice that in this case, all learning is relegated to the parameters of the linear layer and the non-linearity required for classification is provided by the DNPUs. It is the high non-linearity of the DNPU that allows the same performance than the ANN-7D network but with fewer parameters.

These results show that we can combine DNPUs with linear layers to solve high-dimensional classification problems with similar accuracy to standard neural networks with comparable architectures. If implemented in hardware, the fundamental advantage of these novel architectures would come from a reduction in the number of parameters and operations required for a specific performance. Since the projected energy efficiency of DNPUs is similar to that of memristors [26], the observed gain in computation translates to gains in energy consumption. Hence, DNPUs complement memristor crossbar arrays to realise more efficient neural-network emulators completely in hardware. This motivates further exploration of DNPU network architectures.

6.6 Discussion

Motivated by recent advances in nano-electronics, we have introduced tuneable dopant network processing units (DNPUs) as promising candidates for nodes in hardware neural-network emulators, due to their high theoretical throughput and low energy consumption. We have demonstrated that a single DNPU has a capacity comparable to that of a small neural network with one hidden layer, while needing fewer learnable parameters. Furthermore, we have expanded previous work on a single DNPU to a multi-node framework by training and implementing a feed-forward DNPU network with five nodes. By time-multiplexing a single DNPU, this network achieves an accuracy of 94% on a linearly non-separable binary classification task that is too challenging for a single DNPU. Our results show that DNPU networks and the proposed training method give robust solutions to non-trivial ML tasks using noisy hardware elements, a condition prevalent in neuromorphic hardware. Finally, we have shown by simulation that DNPUs allow the classification of MNIST hand-written digits with over 96% accuracy using a network with only 10 DNPUs. To the extent of our knowledge, the examples presented here are the first realisations of non-biological neural networks with high-capacity neurons. As shown here, this approach has the potential of reducing the computational cost in neural networks by reducing the number of operations and parameters required for a given classification performance.

Although the use of DNN models mitigate the device-to-device variation there is an extra overhead when training DNPU networks off-chip, which becomes relevant for large DNPU networks. Our models were not optimized for memory and computational costs. However, there are various considerations to mitigate this. The overhead can be significantly reduced by a systematic exploration of other architectures [36] and using pruning methods to reduce the complexity of the network [15]. Given the recent support in PyTorch for parallelizing the computational graph, we are exploring this approach for large DNPU networks to mitigate the computational footprint. Finally, we have developed a method for on-chip training via gradient descent in-materio [37] that bypasses surrogate models and accounts for device-to-device variation and the noise in the output current.

6.6.1 Towards Large-Scale Hardware Neural-Network Emulators.

Expanding on the examples presented in this paper, we envision the development of large-scale DNPU networks for efficient neural information processing, making use of the unique feature of DNPUs to perform non-linear computations that are commonly only performed in a neural network by combining arithmetic operations and non-linear activation functions. First, by leveraging the expected high throughput, virtual architectures can be implemented by time-multiplexing several independent DNPU devices in parallel. Second, integrating DNPUs with memristor crossbar arrays would allow the co-allocation of memory and computation by implementing connectivity in-materio. Third, direct feed-forward interconnectivity of DNPUs in large circuits would

provide a flexible neural network emulator. The advantage of this approach is that the learned control voltages can be applied directly to the hardware circuit for inference and, in principle, the same hardware can be deployed for various tasks by switching to different control voltage configurations. Furthermore, since computations in all three approaches take advantage of the physical properties of the materials, using DNPUs may reduce the parameters and operations required for inference.

To take advantage of this computational gain, it is necessary to develop DNPU technology to a system level where DNPUs are embedded in an optimized peripheral circuitry. For now, DNPUs work best at 77 K but room temperature operation has been demonstrated [26]. Hence, it is necessary to work on novel DNPU designs to achieve efficient operation at room temperature. This will allow optimizing the peripheral circuitry and increase bandwidth to achieve an energy efficiency similar to that of memristors. We point out that the estimate of the DNPU's efficiency in [26] was conservative. We believe that there is room for further improvement, but what we can aim for will ultimately depend on the particular application, the required accuracy, and implementation of the peripheral electronics and further development of the DNPUs. Finally, an important aspect of our approach for future research is the established off-chip training procedure, which makes research on neural network architectures with high-capacity neurons possible and allows the design of ad-hoc hardware solutions to emulate neural network functionality.

References

[1] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y Shi. Scaling for edge inference of deep neural networks. *Nature Electronics*, 1(4):216–222, 2018. doi: 10.1038/s41928-018-0059-3.

[2] M. Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014. doi: 10.1109/ISSCC.2014.6757323.

[3] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.

[4] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang. A survey of fpga-based neural network accelerator. *arXiv*, 2017. doi: 10.48550/arXiv.1712.08934.

[5] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. Horowitz. Convolution engine: balancing efficiency & flexibility in specialized computing. *ACM SIGARCH Computer Architecture News*, 41(3):24–35, 2013. doi: 10.1145/2735841.

[6] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam. Diannao family: energy-efficient hardware accelerators for machine learning. *Communications of the ACM*, 59(11):105–112, 2016. doi: 10.1145/2996864.

[7] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254. IEEE, 2016. doi: 10.1145/3007787.3001163.

[8] S. Zhang, Z. Du, L. Zhang, H. Lan, S.Liu, L. Li, Q. Guo, T. Chen, and Y. Chen. Cambricon-x: An accelerator for sparse neural networks. *2016 49th Annual IEEE/ACM International Symposium Microarchitecture (MICRO)*, pages 1–12, 2016.

[9] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos. Convlutin: Ineffectual-neuron-free deep neural network computing. *ACM SIGARCH Computer Architecture News*, 44(3):1–13, 2016. doi: 10.1145/3007787.3001138.

[10] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke,

A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12. IEEE, 2017. doi: 10.1145/3079856.3080246.

[11] E. Nurvitadhi, J. Cook, A. Mishra, D. Marr, K. Nealis, P. Colangelo, A. Ling, D. Capalija, U. Aydonat, A. Dasu, and S. Shumarayev. In-package domain-specific asics for intel® stratix® 10 fpgas: A case study of accelerating deep learning using tensortile asic. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 106–1064. IEEE, 2018. doi: 10.1145/3174243.3174966.

[12] V. Vanhoucke, A. Senior, and M. Z. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[13] M. Courbariaux, Y. Bengio, and J. P. David. Training deep neural networks with low precision multiplications. *arXiv*, 2014. doi: 10.48550/arXiv.1412.7024.

[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[15] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv*, 2015. doi: 10.48550/arXiv.1510.00149.

[16] H. S. P. Wong and S. Salahuddin. Memory leads the way to better computing. *Nature Nanotechnology*, 10(3):191, 2015. doi: 10.1038/nnano.2015.29.

[17] E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga. Analog memristive synapse in spiking networks implementing unsupervised learning. *Frontiers in neuroscience*, 10:482, 2016. doi: 10.3389/fnins.2016.00482.

[18] D. Ielmini and H. S. P. Wong. In-memory computing with resistive switching devices. *Nature Electronics*, 1(6):333–343, 2018. doi: 10.1038/s41928-018-0092-2.

[19] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646, 2020. doi: 10.1038/s41586-020-1942-4.

[20] M. Dale, S. Stepney, J. F. Miller, and M. Trefzer. Reservoir computing in materio: An evaluation of configuration through evolution. In *2016 IEEE symposium*

series on computational intelligence (SSCI), pages 1–8. IEEE, 2016. doi: 10.1109/SSCI.2016.7850170.

[21] H. Broersma, J. F. Miller, and S. Nichele. Computational matter: Evolving computational functions in nanoscale materials. In *Advances in Unconventional Computing*, pages 397–428. Springer, 2017. doi: 10.1007/978-3-319-33921-4_16.

[22] X. He, T. Liu, F. Hadaeghi, and H. Jaeger. Reservoir transfer on analog neuromorphic hardware. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1234–1238. IEEE, 2019. doi: 10.1109/NER.2019.8716891.

[23] S. Kan, K. Nakajima, Y. Takeshima, T. Asai, Y. Kuwahara, and M. Akai-Kasaya. Simple reservoir computing capitalizing on the nonlinear response of materials: Theory and physical implementations. *Physical Review Applied*, 15(2):024030, 2021. doi: 10.1103/PhysRevApplied.15.024030.

[24] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015. doi: 10.1109/TCAD.2015.2474396.

[25] M. Pfeiffer and T. Pfeil. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in neuroscience*, 12:774, 2018. doi: 10.3389/fnins.2018.00774.

[26] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.C. Ruiz Euler, H. Broersma, P.r A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel. Classification with a disordered dopant-atom network in silicon. *Nature*, 577(7790):341–345, 2020. doi: 10.1038/s41586-019-1901-0.

[27] S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature Nanotechnology*, 10(12):1048, 2015. doi: 10.1038/nnano.2015.207.

[28] A. Gidon, T. A. Zolnik, P. Fidzinski, F. Bolduan, A. Papoutsi, P. Poirazi, M. Holtkamp, I.e Vida, and M. E. Larkum. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87, 2020. doi: 10.1126/science.aax6239.

[29] H-C. Ruiz Euler, M. N. Boon, J. T. Wildeboer, B. van de Ven, T. Chen, H. Broersma, P. A. Bobbert, and W. G. van der Wiel. A deep-learning approach to realizing functionality in nanoelectronic devices. *Nature Nanotechnology*, 15 (12):992–998, 2020. doi: 10.1038/s41565-020-00779-y.

[30] J. F. Miller, S. L. Harding, and G. Tufte. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67, 2014. doi: 10.1007/s12065-014-0106-6.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019. doi: 10.5555/3454287.3455008.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. doi: 10.48550/arXiv.1412.6980.

[33] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. ISBN 978-0-387-84857-0.

[34] X. Xu, M. Tan, B. Corcoran, J. Wu, A. Boes, T. G. Nguyen, S. T. Chu, B. E. Little, D. G. Hicks, R. Morandotti, A. Mitchell, and D. J. Moss. 11 tops photonic convolutional accelerator for optical neural networks. *Nature*, 589(7840):44–51, 2021. doi: 10.1038/s41586-020-03063-0.

[35] R. Han, Y. Xiang, P. Huang, Y. Shan, X. Liu, and J. Kang. Flash memory array for efficient implementation of deep neural networks. *Advanced Intelligent Systems*, page 2000161. doi: 10.1002/aisy.202000161.

[36] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019. doi: 10.5555/3322706.3361996.

[37] M. N. Boon, H.C. Ruiz-Euler, T. Chen, B. van de Ven, U. Alegre Ibarra, P. A. Bobbert, and W. G. van der Wiel. Gradient descent in materio. *arXiv*, 2021. doi: 10.48550/arXiv.2105.11233.

7

Dopant Network Processing Units as Tuneable Extreme Learning Machines

Inspired by the highly efficient information processing of the brain, which is based on the chemistry and physics of biological tissue, any material system and its physical properties could in principle be exploited for computation. However, it is not always obvious how to use a material system's computational potential to the fullest. Here, we operate a dopant network processing unit (DNPU) as a tuneable, Extreme Learning Machines (ELM) and combine the principles of evolution in-materio and ELM to optimise its computational performance on a nonlinear classification benchmark task. We find that, for this task, there is an optimal, hybrid operation mode ('tuneable ELM mode') in between the traditional ELM regime with a fixed DNPU and linearly weighted outputs ('fixed-ELM mode') and the regime where the outputs of the nonlinear system are directly tuned to generate the desired output ('direct-output mode'). We show that the tuneable ELM mode reduces the number of parameters needed to perform the formant-based vowel recognition benchmark task. Our results emphasise the power of in-materio computing and underline the importance of designing specialised material systems to optimally utilise their physical properties for computation.

Chapter in preparation as: B. van de Ven, U. Alegre-Ibarra, P. J. Lemieszczuk, P. A. Bobbert, H.-C. Ruiz-Euler, W. G. an der Wiel, Dopant network processing units as tuneable Extreme Learning Machines.

Contributions: Conceiving the project, fabrication, measurements, data analysis and writing

7.1 Introduction

Spurred by the increasing computational demands of artificial intelligence (AI), the difficulties of conventional computing hardware to keep up with these demands, and inspired by the highly efficient information processing of the brain, there is a growing interest in harnessing chemical and physical phenomena in material systems for complex, efficient computations[1]. This growing field of research goes by different names, such as unconventional, natural or in-materio computing[2, 3]. Designing a material system as a computing device is a nontrivial task, especially on the nanoscale. Instead, one can train certain designless material systems to exhibit functionality, a process we refer to as 'material learning'[4], in analogy to 'machine learning' in software systems.

A recent example of such a designless tuneable nanoscale material system that can be trained for functionality using material learning is a dopant network processing unit (DNPU)[4]. It consists of a network of donor or acceptor dopant atoms in a semiconductor host material, where charge carriers hop from one dopant atom to another under the influence of input and control voltages applied at surrounding electrodes. The output consists of the current(s) measured at one or more electrodes. For material learning using DNPUs, their electronic properties should be tuneable, nonlinear, and exhibit negative differential resistance (NDR). This can be realised by varying one or more control voltages, as shown in Figure 7.1a. Among other things, DNPUs have been shown to have the capability of solving nonlinear classification tasks[4]. There are several methods to obtain desired functionality in DNPUs by material learning. We have demonstrated DNPU training with evolution-in-materio[5], off-chip gradient descent[6], and more recently, gradient descent in materio[7].

Evolution-in-materio is a material-learning approach in which (digital) computer-controlled evolution is used to manipulate a physical system. It allows the exploitation of complex physical effects, even if these effects are a priori unknown[8]. Off-chip gradient descent is an approach where an artificial neural network (ANN) is trained to emulate the behaviour of the physical DNPU. This allows the use of use standard deep-learning methods to determine the control voltages that are needed to reach a desired functionality[6]. Gradient descent in materio is a method that uses lock-in techniques to extract the gradient in the output with respect to the tuneable parameters by perturbing these parameters in parallel with different frequencies. Using the extracted gradient, it is possible to gradually move towards a desired functionality directly in the material system[7].

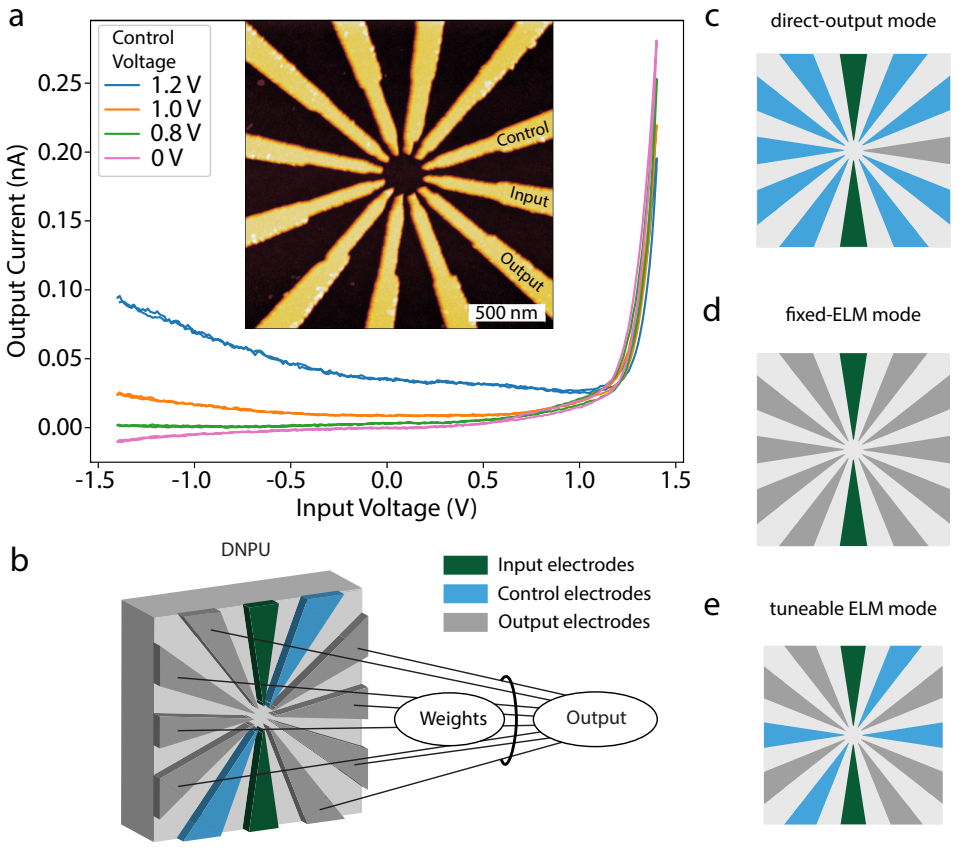


Figure 7.1: Dopant Network Processing Units as tuneable ELM: a IV curves measured between an input and an output electrode of a 12-electrode dopant network processing unit (DNPU), for different control voltages applied to a third control electrode (0 V is applied to the other electrodes), exhibiting negative differential resistance when the control voltage increases (orange and blue curves). Inset: atomic force microscope image of the (DNPU). b Schematic of a DNPU used in the 'tuneable ELM mode', with input (green), output (grey) and control (blue) electrodes. The final output of the system is obtained by linearly combining the output currents with tuneable weights. c Schematic of the conventional, 'direct-output mode'. In this case there is a single output (without linear weight factor), while the other electrodes are either used as input or control. d Schematic of the 'fixed-ELM mode': all electrodes are either inputs or outputs, where the latter are linearly combined to form the output. e Schematic of the 'tuneable ELM mode', which is a hybrid of the modes shown in c and d.

Another popular framework for utilising the complexity of disordered material systems is reservoir computing (RC). Independently developed RC schemes are echo state networks (ESNs)[9], liquid state machines (LSMs)[10], and the backpropagation-decorrelation (BPDC) on-line learning rule[11]. Generally, the concept of RC was used in combination with recurrent neural networks (RNNs), which exhibit time-dependent (dynamic) behaviour[12]. Here, the network (reservoir) projects input values nonlinearly into a high-dimensional space of its reservoir states. To train such a network to perform a certain task, a linear, supervised reservoir readout layer is used to map the reservoir states to a desired output. As only the weights of the readout layer need to be trained, while the random network itself remains fixed during the training process, the training is relatively fast and efficient as compared to other neural network training methods[13]. Its general applicability makes RC a suitable approach to utilise disordered material-based networks to perform desired computations, as has been shown, for example, in network of carbon nanotubes[14] and polymers[15]. Another approach that linearly combines the output states of a network is extreme learning machines (ELM)[16]. Recent work shows that ELM is effectively RC without time dynamics. An opto-electronic network is used to perform both RC and ELM based a switch to turn a feedback loop on or off[17]. Figure 7.1b illustrates a 12-electrode DNPU operated in the tuneable ELM mode, where the black lines indicate the weighted connections of the readout layer mapping the DNPUs output states (currents) to the desired output of the network.

In the present study we specifically consider a DNPU based on boron-doped silicon with a novel geometry having 12 electrodes (see section 7.6). Operating this DNPU in the tuneable ELM mode, we optimise the computational power that can be achieved. This tuneable ELM approach is inspired by the work of Dale et al.[18], who showed that by combining the concept of RC with evolution-in-materio, it is possible to obtain a reservoir capable of reaching higher accuracies for RC benchmark tasks[14, 18]. Their work mainly focusses on the use of micron-scale carbon nanotube / polymer mixtures. DNPUs are different from these systems in several ways. On the one hand, they have a smaller footprint and are silicon-based, which may facilitate scaling and integration with conventional electronics. On the other hand, DNPUs do not exhibit time dynamics at the timescales of our measurements. This makes these systems, in their present form, unsuitable to process data directly in the time domain. However, the lack of dynamical behaviour allows DNPUs to be used with the off-chip gradient descent approach[6]. The advantage of this training technique is that it will allow DNPUs to be incorporated in bigger networks of coupled DNPUs and other ANN elements to create a new combined network that can be trained in one training run, after which the task can be implemented in the material platform. This could lead to a universal training technique allowing for the use of different material platforms (e.g. a DNPU as tuneable nonlinear ELM combined with memristor technology for linear operations) to be optimised at the same time.

We study the computational power of a DNPU in the tuneable ELM operation mode for different numbers of control (N_C) and output (N_O) electrodes, using two input electrodes. To quantify computational power we use the Vapnik-Chervonenkis, VC dimension[19], defined as the maximum number of non-collinear points that can be classified into all possible binary groups. Since, it is a priori, not known how many control and output electrodes are needed to realise the highest computational capability, we study multiple tuneable ELM modes, going from the standard direct-output mode ($N_C = 9$, $N_O = 1$, figure 7.1c) to the fixed-ELM mode ($N_C = 0$, $N_O = 10$, figure 7.1d). As an example, a tuneable ELM mode with $N_C = 4$, $N_O = 6$ is schematically represented in Figure 7.1e. First, we search for the optimal operation mode for performing binary classification tasks. With the best operation mode, we demonstrate the use of the off-chip gradient descent training technique[6] to perform the formant-based vowel recognition benchmark task[20] with up to 5 emulated DNPUs in parallel. Finally, we will use the vowel recognition benchmark to show that using DNPUs as tuneable ELMs allows one to reduce the number of parameters that need to be tuned and stored as compared to an ANN counterpart. Our results emphasise the power of in-materio computing and underline the importance of combining different material platforms in a way that optimises their computational capabilities.

7.2 VC Dimension Analysis

In Figure 7.2a the measurement scheme of the VC dimension analysis is presented. To perform this analysis, 7 input points are defined in a two-dimensional voltage input space (as represented by the two waveforms in the left part of Figure 7.2a). These points are chosen within the working range of -1.1 to 1.1 V of the DNPU that we used for our study. The first four input points are chosen such that for the fixed-ELM mode all binary classifications can be realised. The other three points are chosen in a way similar to Ruiz-Euler et al.[19], as explained in section 7.7.1. Using these input points, the complexity of the task can be varied by the number of points that need to be correctly labelled by the tuneable DNPU ELM and the linear layer (Figure 7.2a, middle part), where a perceptron is used to determine when the points are correctly labelled (see section 7.6). For each n input points there are 2n possible labels. In Figure 7.2a, the DNPU and the linear layer are trained to yield as output the binary label [0000110], which is one of the $2^7 = 128$ labels for VC dimension 7.

The training of the DNPU is performed using a combination of a computer-assisted genetic algorithm (GA)[21] (to find the voltages needed to tune the DNPU) and pseudoinverse learning[22] (to find the optimal weights in the linear readout layer). For each genome (set of control voltages) in the genetic algorithm, new weights for the readout layer are found using pseudoinverse learning after which the output of the network is mapped to a class probability using a perceptron (see section 7.6). For VC dimension 6 (blue points) and 7 (orange points) several different tuneable ELM modes have been studied, see Figure 7.2b: $N_C = 0$, 1, 2, 4, 6 and 9 for VC dimension 6. The

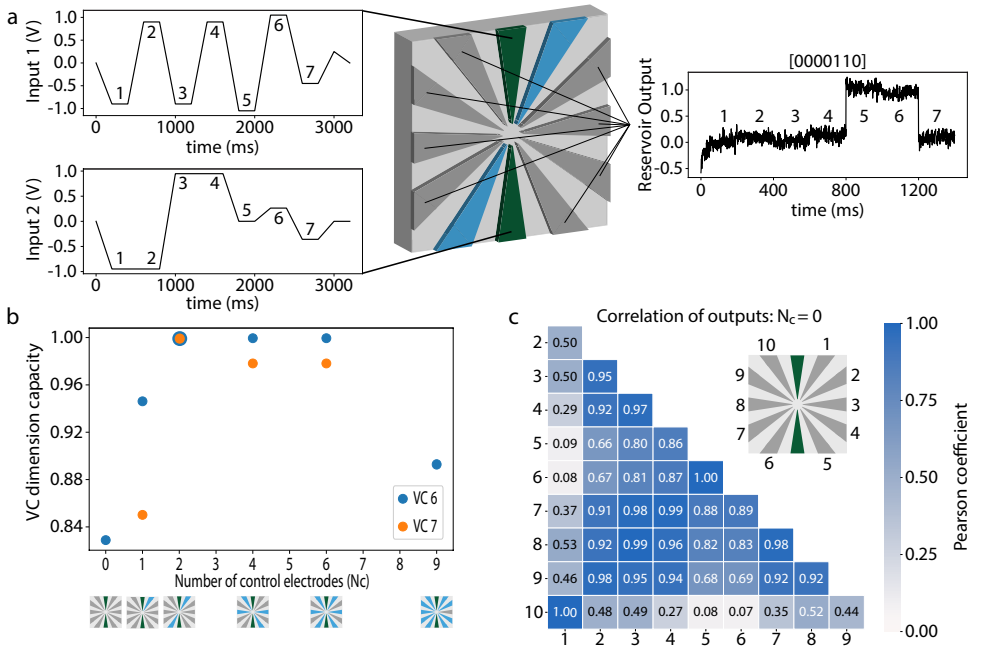


Figure 7.2: VC dimension analysis: a Schematic of the VC dimension benchmark for the raw input waveforms on the left using the DNPU (inputs: green, controls: blue, outputs: grey) and the readout layer in a tuneable ELM operation mode, with $N_C = 2$ control electrodes and $N_O = 8$ output electrodes (middle). The output waveform of label [0000110] for VC dimension 7 is displayed on the right, where only the output at the plateaus of the input is shown. b The capacity (fraction of correct labels) for the different operation modes, indicated by the number of control electrodes used. Blue: VC dimension 6 (6 input points). Orange: VC dimension 7 (7 input points, only for the tuneable ELM modes). The DNPU schematics at the x-axis show the electrode configurations. c Correlation matrix of the output currents of the DNPU for the fixed-ELM mode ($N_C = 0$) for VC dimension 6. The darker the blue, the higher the correlation in the output currents.

$N_C = 0$ and $N_C = 9$ cases are the direct output and fixed-ELM modes, respectively. All values of NC in between correspond to the tuneable ELM modes. Because for VC dimension 6 the tuneable ELM modes have the highest computational capacity, we only analyse these further for VC dimension 7. We observe that the $N_C = 2$ mode has the highest computational capability since this operation mode has the highest capacity for VC dimension 7.

The fixed-ELM mode underperforms because some DNPU outputs highly correlate. It is well-known that the linear separation capacity of a specific ELM network grows with the number of independent output states (output channels)[23]. Consequently, highly correlated ELM states reduce the expressivity of the ELM. To investigate this behaviour, we analyse the correlations in the DNPUs output states. This is done by calculating the Pearson correlation coefficient between all the output states of the fixed-ELM mode, as shown in Figure 7.2c. It is observed that most of the correlations between the outputs are relatively high (dark blue); see also section 7.7.1. Outputs 1 and 10 as well as 5 and 6 even have a correlation coefficient of 1.00, meaning that they provide the same information. Therefore, using output electrodes 1 and 6 instead as control electrodes does not result in a loss of information, while allowing to tune the DNPU towards a more complex output response. Based on this observation, the choice of which electrodes to use as control for the $N_C = 1$, 2 modes is straightforward. For the other two tuneable ELM modes ($N_C = 4$, 6) the control electrodes were chosen such that at least one output or input electrode is in between the control electrodes resulting in the DNPU schematics in Figure 7.2b. To further illustrate the choice of the electrodes for the $N_C = 1$, 2 modes we show the correlation matrices for $N_C = 2$ for two different labels in Figure 7.5a and b. The reason why the $N_C = 2$ mode performs better than the fixed-ELM mode is that correlation coefficients very close to 1.00 do not occur anymore.

7.3 Vowel Recognition Using Tuneable DNPU ELMs

To demonstrate the capability of the DNPU to perform more complex tasks, we focus on the Hillenbrand formant-based vowel recognition benchmark task[20]. For this task, we emulate the behaviour of multiple DNPUs in parallel (the behaviour of all DNPUs is derived from a single physical DNPU). Formant-based vowel recognition is a classification benchmark with a limited number of features and classes, making it a task that can be solved with a limited number of DNPUs. Hillenbrand et al.[20] extracted the formants from recordings of a spoken vowel at different times, which are broad spectral acoustic maxima caused by acoustic resonances in the human vocal tract. This allows the transformation of a task that is commonly performed using dynamic systems to a static benchmark task, making it compatible with DNPUs. Adult male/female, as well as boy/girl speakers, each pronounced 12 different vowels. From the recordings, a dataset is constructed by first extracting the fundamental frequency or pitch f_0, which is the lowest frequency present in a spoken vowel. This frequency is directly linked to the size of the speaker's vocal cords. Therefore, men tend to have a low f_0 and women a high f_0. This is a useful measure to take into account when classifying over different ages and genders, as is done in this benchmark[24].

After determining f_0, Hillenbrand et al. extracted 4 formants before the vowel is spoken (known as the "steady state") and 3 formants at 20%, 50% and 80% of the spoken vowel duration. In Figure 7.3a we show a spectrogram of the vowel "oa"

pronounced by an adult male, where the vertical features indicate the different formant frequencies and the horizontal lines the times at which they have been extracted. This results in a total of 14 features to be used as the inputs for the task.

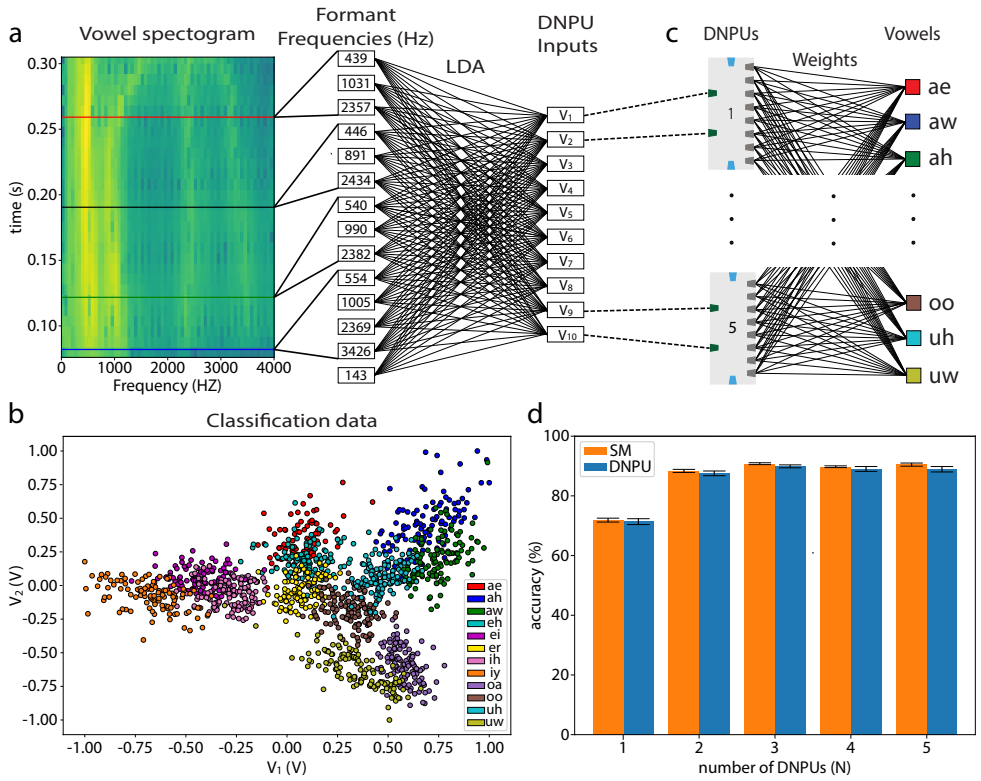


Figure 7.3: Spoken vowel recognition task: a Left: spectrogram of an adult male pronouncing "oa". The horizontal lines indicate the times at which the formants are extracted. Right: Mapping of formant frequencies to voltages $V_1 - V_{10}$ using linear discriminant analysis (LDA). b V_2 versus V_1 for all 12 vowels, showing overlap, especially for V_2. c Vowel classification by N DNPUs in parallel, with a linear readout layer: V_1 and V_2 are inputs to DNPU1, V_3 and V_4 are inputs to DNPU2, etc. d Classification accuracies for N clones of the physical DNPU and of a surrogate model (SM) of the DNPU.

To use a single DNPU in the $N_C = 2$ tuneable ELM mode, we need flexibility in the number of inputs. To reduce this number, while still using the information from all 14 features, we use a linear discriminant analysis (LDA), where the input data are linearly mapped to a feature space where classes have the highest separation (between-class covariance) and lowest within-class covariance. After this linear mapping, the input

data are normalised to the DNPU input range (-1.0 to $1.0\ V$). This results in 10 new inputs ordered from the best (V_1) to the worst (V_{10}) classifying input, see Figure 7.3a. In Figure 7.3b, V_2 is plotted against V_1. It is seen that the vowels have less overlap when projected to V_1 than to V_2, as is expected when using LDA. To demonstrate the computational capabilities of the DNPUs without needing to fabricate several of them, we use the single DNPU analysed above and measure its output currents when either V_1 and V_2, V_3 and V_4, etc., are applied as input voltages. This corresponds to putting $N = 1 - 5$ identical 'cloned' DNPUs in parallel. Effectively, this corresponds to using 5 different DNPUs, because the control voltages of the cloned DNPUs will be different, leading to completely different input-output characteristics (in the next paragraph we explain how the control voltages are determined). For the cases that $N < 5$ the last $2 \cdot (5 - N)$ inputs are discarded. Finally, we use a single linear readout layer with as inputs the measured currents, as stored in a digital computer, of the N times cloned DNPU; see Figure 7.3c. We call this method of performing the vowel recognition task, where we use a physical device in combination with operations performed in a computer, a 'hybrid' method. We note that the measured currents contain noise and other uncertainties present in a fully physical implementation.

In addition to the network with the cloned physical DNPU, a network is created with N clones of an ANN surrogate model (SM) of the DNPU that accurately emulates its behaviour[6]. To train this network for the vowel recognition task, we use the off-chip gradient descent-based training technique we introduced in Ref. [6]. We extend that work by including the linear readout layer in a complete ANN model of the full network (the SM and the linear readout layer; see Figure 7.3c). This allows us to use standard deep-learning methods to train the complete network for the vowel recognition task. The found N pairs of control voltages are applied to the physical DNPU, after which the vowel recognition is performed with the hybrid method.

Figure 7.3d shows the accuracies in the vowel recognition after the training, both for the networks with N clones using the DNPU SM (orange, SM method) and the physical DNPU (blue, hybrid method). For each N, we performed 6 training runs (see section 7.6), of which we chose, the 5 attempts with the highest recognition accuracy to remove potential outliers. The average values and standard deviations of these 5 attempts are used to obtain the reported accuracies and error bars. We observe an increase in average accuracy from 71.4% to 89.9% for the hybrid method between $N = 1$ and $N = 3$. The saturation in accuracy when N is further increased, is attributed to the limited added information in $V_7 - V_{10}$. This can be directly linked to the LDA method, where the first LDA components have the largest linear separability (least overlap) and the last LDA components the smallest. We elaborate on this in section 7.7.2 and illustrate this in Figure 7.6.

We also observe in Figure 7.3d that transferring the control voltages found in the SM method to the hybrid method results in very limited performance loss (from 90.6% to 89.9% for $N = 3$). This shows the robustness of the SM method. The small difference

is attributed to remaining model uncertainties in the SM and the inability of the SM to account for noise. In Figure 7.7, we show that when replacing the LDA by a trainable linear layer with 14 inputs and $2N$ outputs in a further extended ANN model, we can increase the highest accuracy obtained in the vowel recognition task from 90.6% to 92.6% for the SM method. A similar improvement is expected for the hybrid method.

7.4 Parameter Reduction Using DNPU ELMs

In this subsection, we show that the number of tuneable parameters needed in the vowel recognition task using DNPUs is, for a comparable accuracy, much less than using ANNs. We consider the network structure discussed in the previous subsection; see Figure 7.3. We use ANNs with 2 inputs and 8 outputs, equal to those of the DNPU in the optimal tuneable ELM mode ($N_C = 2$, $N_O = 8$; see Figure 7.2). To reach a comparable recognition accuracy as with (cloned) DNPUs, the ANNs can be much smaller than the ANN used in the surrogate model (SM) of the DNPU. We use as ANN structure a fully-connected nonlinear network between the inputs and the outputs with a ReLU activation function (see section 7.6). Using $N = 1 - 5$ of these small ANNs, illustrated in Figure 7.4a with the red boxes, we make a network with the same structure as that in Figure 7.3, but with the ANNs replacing the (cloned) DNPUs. We train this network and use it to perform the vowel recognition task from the previous subsection. Figure 7.4b compares the achieved average accuracy and error (green) to the ones achieved using the DNPU (blue). The accuracy and error were obtained in the same way as in the previous subsection (extracted from 5 of the 6 training runs). We see that, especially for high N, the average accuracies of the hybrid measurements using cloned DNPUs are quite close to those of the ANNs (89.9% compared to 91.4% for $N = 3$).

Since one of the important limitations of deep learning is the storage and retrieval of the values of the parameters used in an ANN[25], it is important to reduce the number of parameters. For an ANN, the number of parameters is equal to the number of weights and biases. In Figure 7.4c, the number of parameters (weights + biases for the ANNs and control voltages for the DNPU) of the ANN replacing the DNPU in the network is plotted against N (green: $24N$). For the DNPU case, the number of parameters is equal to the number of control voltages (blue: $2N$). The reason why the number of parameters for the DNPU case is smaller than that for the ANN case is that the control voltages of the DNPU have a global effect on the outputs[4]. While for the DNPU one parameter influences all outputs, for the ANN one parameter influences only one output. We note that in the consideration of the number of parameters, we did not take account the weights in the linear readout layer. The reason is that we want to focus on the capabilities of the DNPU itself, which can be exploited in various other tasks than only vowel recognition. We conclude that their global tuning capability in combination with their non-linear input-output relation

can make DNPUs a powerful tool for in-materio computation.

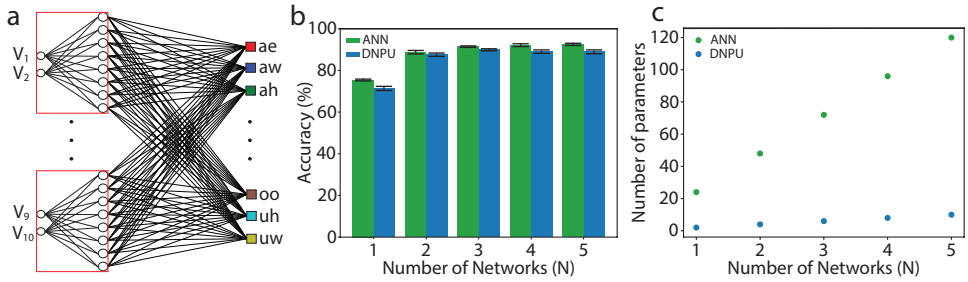


Figure 7.4: Parameter reduction using DNPUs: (a) Same network as in Figure 7.3c, but with the DNPUs replaced by single-layer fully-connected ANNs (red boxes). (b) Vowel recognition accuracy for the network with ANNs (green) as compared against the network with DNPUs (blue, same data as in Figure 7.3d). (c) Number of tuneable parameters for the network with ANNs (green) compared against the network with DNPUs (blue).

7.5 Discussion

We have studied the computational power of a dopant network processing unit (DNPU), consisting of a region with boron dopants at the surface of a silicon substrate, contacted with 12 electrodes that can be used as data inputs, controls and outputs. We used the complex nonlinear dependence between input voltages and output currents of the DNPU, tuned by voltages applied to the control electrodes, in combination with a linear readout layer for ELM. Three modes of operation were considered: (1) the 'fixed-ELM mode', (2) the 'direct-output mode' and (3) the 'tuneable ELM mode'. In the fixed-ELM mode (1), where the DNPU cannot be tuned, all electrodes are used as inputs or outputs. In the direct-output mode (2), the electrodes are used as inputs, tuneable controls and one or more outputs, without using a readout layer. The tuneable ELM mode (3) is a combination of (1) and (2), where the DNPU can be partially tuned by control voltages. We found that the tuneable ELM mode provides the highest computational power, as quantified by the Vapnik-Chervonenkis (VC) dimension. For the case of 2 data inputs, we found optimal computation power with 2 controls and 8 outputs. The fact that the fixed-ELM mode (10 outputs) has suboptimal computing power was rationalised by considering the output correlation matrix, which shows strong correlations between the two outputs adjacent to the each of the two inputs. In that case the computation power can be increased by using one of these outputs to as a control electrode, leading to a total of 2 controls. We conclude from this analysis that consideration of the output correlation matrix is important for optimizing the operation mode of a complex tuneable ELM.

We solved the vowel recognition benchmark task by a hybrid network of emulated DNPUs in the optimal tuneable ELM mode in combination with a common linear readout layer, where the dimension of the input data are reduced while maximising the class separation (using LDA). As training method, an off-chip gradient descent method was applied using a surrogate model (SM) of the physical DNPU, yielding pairs of control voltages that we applied to the physical DNPU for validation. The noise present in the physical DNPU and deviations from the SM lead to only a small decrease in the accuracy of the vowel recognition task in this hybrid method, as compared to using the SM. Since the different pairs of control voltages in the hybrid method effectively correspond to using different DNPUs, there should be equivalence with a network using different physical DNPUs. It should therefore be possible to extend this approach to large and fully physical networks with a large number of DNPUs, where also the LDA and the linear readout layer are realised by physical systems such as memristors[26] and optical networks[27]. Since training of the latter systems is also done by physically implementing the parameters found by artificial neural networks (ANNs)[26, 27], it should be possible to incorporate the training of these physical networks in the off-chip training technique in a similar way as demonstrated in the present work. This could potentially lead to physical networks trained for classification tasks that outperform ANNs regarding inference. We showed that the global, nonlinear tuneability of a DNPU requires fewer parameters than a minimal ANN that has similar input-output behaviour. Since these parameters should be stored in memory and retrieved for each recognition task, the use of DNPUs instead of ANNs will be less memory-intensive.

The vowel recognition task demonstrated in this work was performed with an accuracy of 89.9% on the test dataset by emulating the behaviour of multiple DNPUs from one physical DNPU. An increase to 92.6% accuracy was shown to be possible by training the DNPU and all linear layers instead of employing LDA. These accuracies are similar to those reported in other work: 89% using spin-torque nano-oscillator[28] and 93% accuracy using an optical system[29]. It should be noted however, that in both these approaches a subset of the data was used that only included female speakers and 7 vowels making the task easier to solve. Since we include both female and male voices of adults and children on all 12 vowels, we conclude that our work presents a neuromorphic network approach with a high classification accuracy on the full vowel recognition benchmark.

The present work shows the power of using designless disordered systems, such as DNPUs, for computation and provides insight into optimally harnessing their computation power. Building on this, we have indicated how such systems can be combined with other physical systems, exploiting each system's optimal operation type (such as linear vs non-linear), to create networks achieving in-materio computation that is fast as well as energy and space efficient.

7.6 Appendix A: Methods

7.6.1 The Dopant Network Processing Unit

The DNPU used in this work is fabricated in a similar fashion as the one in Chen et.al.[4]. The main difference is the number of electrodes: 12 instead of 8. These electrodes are made by e-beam evaporation of 1 nm Ti and 25 nm Pd and placed on top of the boron doped silicon in a circle with a diameter of 300 nm. The boron concentration under the contacts is approximately $2 \cdot 10^{19}$ cm^{-3}, which creates an ohmic contact between the electrodes and the substrate. Using the electrodes as a mask, the silicon is etched such that the boron concentration at the receded silicon surface is reduced to approximately $5 \cdot 10^{17}$ cm^{-3}, resulting in variable-range hopping at 77 K[4].

7.6.2 Measurements

All DNPU measurements are performed using a customised dipstick to insert the device into liquid nitrogen (77 K). To measure its behaviour, the DNPU is wire bonded to a printed circuit board (PCB) that has 12 IV converters connected, allowing us to use up to 12 output channels. These IV convertors have either 10 MΩ or 100 MΩ feedback resistances allowing us to measure a current range of -400 nA to 400 nA or -40 nA to 40 nA, respectively. Output electrodes that are close to the input/control electrodes have relatively large output currents and are connected to the 10 MΩ IV converters. This allows measuring the full output current range. The electrodes further away from the input/control electrodes are connected to the 100 MΩ IV converters, such that the relatively small output currents can be determined more accurately. We calculate the measured voltages back to current values in nA. During the measurements, the voltages are applied and measured using a national instruments compactDAQ (NI cDAQ-9132) with two modules, one for digital-to-analogue conversion (NI-9264) and one for analogue-to-digital conversion (NI-9202). This measurement system is automated using Python (https://github.com/BraiNEdarwin).

7.6.3 Vapnik Chervonenkis Dimension Training

To train the network (DNPU + linear readout layer) to yield all the binary labels of the corresponding VC dimension n we combine a genetic algorithm[21] (GA) with pseudoinverse learning[22]. This is done by randomly initialising a set of 25 genomes of control voltages in the range -1.1 to 1.1 V as optimisable parameters. For each genome the current waveforms of all the NO outputs of the DNPU are measured for the input waveforms given in Figure 7.2a. This yields an array X of size $n \cdot N_O$ that are linearly combined to obtain the target output waveforms. The N_O weights W of this layer are calculated using the pseudo-inverse learning method, which solves[22]:

$$W = Y_t \cdot X^+ \tag{7.1}$$

where Y_t is the target output waveform (consisting of a sequence of 0's and 1's) and X^+ is the pseudoinverse of X. The 25 obtained network output waveforms $Y = W \cdot X$ are evaluated and ranked based on their fitness (F). In this work we define the fitness to be the correlation between the actual and the target output waveforms (Y and Y_t) multiplied by a sigmoid function of the sep between the lowest high (binary 1) and the highest low (binary 0) state, see equation 7.2[6]:

$$F = corr(Y, Y_t) \cdot \frac{1}{1 + e^{-2 \cdot (sep - 2)}} \tag{7.2}$$

Based on the fitness ranking of the present genomes, the next generation of genomes is spawned by: (1) keeping the 5 best genomes from the previous generation; (2) slightly altering the 5 best genomes by introducing small fluctuations to the genes (voltages); (3) generating 5 genomes by cross-breeding (blend alpha beta crossover, BLX-alpha-beta) between the 5 best performing genomes with a certain probability; (4) 5 genomes are obtained via crossbreeding between the top 5 and 5 random genomes; (5) the last 5 genomes are randomly generated. For details, we refer to https://github.com/BraiNEdarwin. The GA is performed for 25 generations.

After the GA optimization, the combination of control voltages corresponding to the genome with the highest fitness is applied to the DNPU. To determine whether the correct labels are obtained by the DNPU and the linear readout layer we use a perceptron. We train the perceptron with the output waveforms and the correct labels. The perceptron is trained for 200 epochs using adaptive moment estimation (Adam) gradient descent for 200 epochs using a binary cross-entropy loss function[30], with a learning rate of $7 \cdot 10^{-3}$, beta coefficients for running averages of (0.9, 0.999), a numerical stability constant $\epsilon = 10^{-3}$, and a weight decay of 0. After training the perceptron, the accuracy is determined by calculating the correctly obtained labels divided by the total number of labels, where a label is noted as found when 100% accuracy is achieved. we give a total of two attempts of running the whole algorithm to each label. A label is correctly classified/found when an accuracy of 100% is reached in one of the two attempts.

7.6.4 Hillenbrand Dataset

The Hillenbrand dataset consists of 12 vowels spoken by male, female, boy and girl speakers. For each of these speakers the duration of the vowel and the fundamental frequency is stored. Besides the fundamental frequency there are 13 formants extracted per speaker, 4 at the steady state (before speaking) and 3 at 20%, 50% and 80% of the spoken vowel. This has been done for $1,669$ recordings. In some cases not all formants could be determined. In these cases the frequency of the formant is denoted as a zero. We removed these samples from the dataset, which leaves us with $1,373$ recordings. For our analysis we do not consider the duration of the recording. We map the formant data to voltages using a linear weight matrix, normalising the voltages such that they lie in the DNPUs voltage range ($-1.1\ V$, $1.1\ V$). The weight

matrix is generated using the linear discriminant analysis (LDA) function from the Python sklearn library[31].

7.6.5 Off-Chip Gradient Descent

The off-chip gradient descent method uses an artificial neural network (ANN) trained on the input-output data of the DNPU. This is done by following the approach used by Ruiz-Euler et al.[6], where the number of activation electrodes (input + control electrodes) for the case $N_c = 2$, $N_o = 8$ is reduced from 7 to 4 and the number of output electrodes increased from 1 to 8. We take $4,850,000$ samples to train the ANN. This ANN consists 5 fully connected hidden layers, each with 90 nodes and ReLU activation functions. The ANN with its weights and biases forms the surrogate model (SM) of the DNPU.

Next, we combine $N = 1 - 5$ cloned SMs with a fully connected linear readout layer (Figure 7.3c) and train the combined network for the vowel recognition task using gradient descent with respect to the optimisable parameters, which are the $2N$ SM control voltages and the weights and biases of the readout layer. The internal parameters of the SM are kept constant. For the vowel recognition training, the $1,373$ recordings from which the formants are extracted are separated into train, validation and test data (861, 256 and 256 recordings, respectively). The validation dataset is used to prevent overfitting on the training dataset. After each training epoch, the found set of parameters is only saved if the loss function for the validation data is also lower than for the previous set of parameters, giving as a final result the set of parameters with the lowest validation loss. For the calculation of the losses, the cross-entropy loss function is implemented using pytorch[30]. The parameter optimization was done using Adam for 500 epochs, with a learning rate of $5 \cdot 10^{-2}$, beta coefficients for running averages of (0.9, 0.999), a numerical stability constant of $1 \cdot 10^{-8}$, and a weight decay of 0. The results reported in the main text are obtained for the test dataset using the final parameters. In total, 6 different random initialisations and divisions of the training and validation data are used for training, keeping the test data unchanged.

In the comparison of the number of parameters (Figure 7.4), the small ANNs, with 2 inputs and 8 outputs, have 16 weights and 8 biases. These small ANNs are incorporated in the network described above, where each cloned DNPU SM is replaced by a small ANN. The training of this network occurs in exactly the same way as the network with the cloned SMs, where the $16N$ weights and $8N$ biases take over the role of the $2N$ DNPU control voltages.

7.7 Appendix B: Supplementary Information

7.7.1 Correlation Analysis

In Figures 7.5a and 7.5b the correlation matrices of the output currents for the classification tasks 5 ([0 0 0 0 1 0 1]) and 6 ([0 0 0 0 1 1 0]) are shown for the $N_C = 2$, $N_O = 8$ mode after training. In contrast to Figure 7.2c in the main text, no correlation coefficients very close to 1.00 occur, while even negative correlation coefficients occur. This explains why the $N_C = 2$ mode has a higher computational power than the $N_C = 0$ mode of Figure 7.2c in the main text.

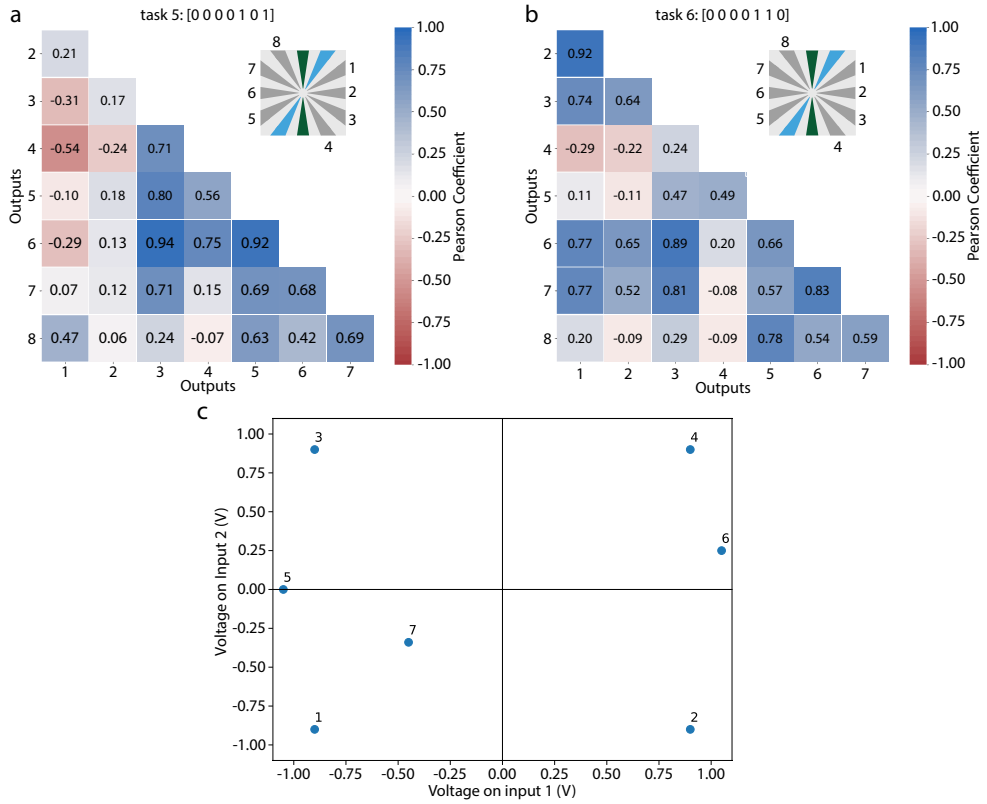


Figure 7.5: a Correlation plot for label 5 ([0 0 0 0 1 0 1]) of the outputs of the DNPU in the $N_C = 2$ mode. b Correlation plot for label 6 ([0 0 0 0 1 1 0]) of the outputs of the DNPU in the $N_C = 2$ mode. c Input points used for the VC dimension analysis presented in the voltage input space.

In Figure 7.5c the voltages V_{in1}, V_{in2} used for these input combinations are shown. As mentioned in the main text, points $1-4$ are chosen within the voltage range ($-1.1\ V$, $1.1\ V$) and optimised for the fixed-ELM mode. To avoid collinearity, the other points are chosen by placing them collinearly on the line connecting two points and shifting them a little away from this line to lift collinearity. Point 5 is placed in between point 1 and 3 and shifted along the V_{in1} input. Point 6 is placed in between points 2 and 4 and shifted along the V_{in2} axis. Point 7 is placed in between points 1 and 4 and shifted both along the V_{in1} and V_{in2} axis. The shifts are chosen in a similar way as in Ruiz-Euler. et. al.[19] and such, that a good spread of points is obtained.

7.7.2 Linear Discriminant Analysis and Classification

To perform the formant-based vowel recognition task, the linear discriminant analysis (LDA) method was used to map the 14 formant features to feature space of reduced dimensionality, while maximizing the linear separability of the classes. By normalising these values, we obtain 10 DNPU input voltages $V_1 - V_{10}$. The LDA involves one mapping that is kept constant, so that we stay closer to the ELM approach, where

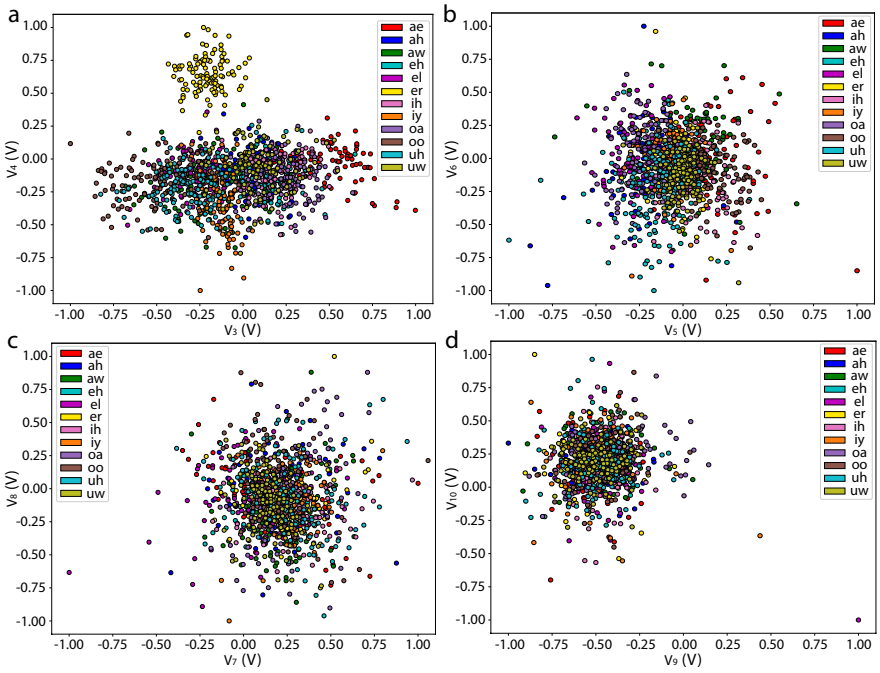


Figure 7.6: a Inputs and corresponding vowel labels for the second (cloned) DNPU. b Same, for the third (cloned) DNPU. c Same, for the fourth (cloned) DNPU. d Same, for the fifth (cloned) DNPU.

inputs to the ELM are kept fixed during training). As mentioned in the main text, $V_1 - V_{10}$ are ordered from the best (V_1) to the worst (V_{10}) least meaningful information to separate the classes. Figure 7.3b in the main text shows V_2 vs V_1 for all 12 vowels, which are used as inputs to the first (cloned) DNPU. Figure 7.5 shows similar plots for the other voltages (a: V_4 vs V_2, inputs to the second (cloned) DNPU, b: V_6 vs V_5, inputs to the third (cloned) DNPU, c: V_8 vs V_7, inputs to the fourth (cloned) DNPU, d: V_{10} vs V_9, inputs to the fifth (cloned) DNPU). As is clearly seen, the data separability in the output of the LDA layer of V_i gradually decreases with I as can be seen from the overlapping sets of labels.

7.7.3 DNPUs as Tuneable Activation Functions

It is also possible to replace the LDA layer by a trainable linear layer. Using the off-chip gradient descent-based training method, this can be achieved by increasing the size of the combined ANN model and treating the formant features as the input of this combined ANN model, where the DNPU acts as tuneable activation function. In this case L2 regularization is used to keep the input voltages inside the DNPU voltage range (in the rare case that an input voltage falls outside the DNPU voltage range the output is classified as false). This alternative method is applied with the surrogate model (SM) of the DNPU. Figure 7.7 gives classification accuracies for this alternative method using N (cloned) DNPU SMs that can be compared to the

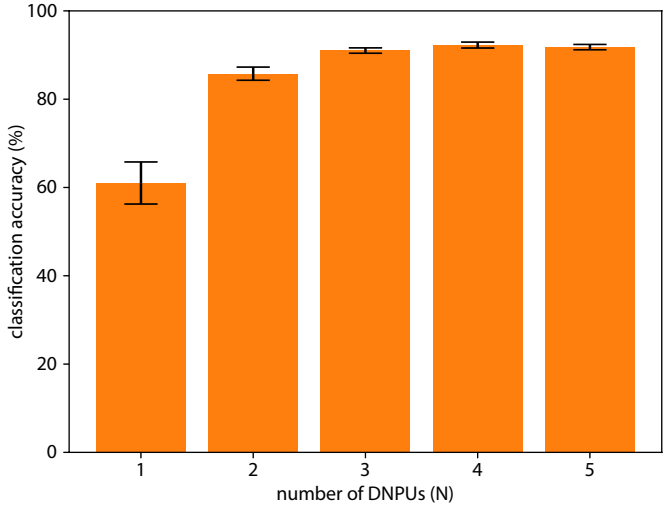


Figure 7.7: Classification accuracies of the vowel recognition task using various numbers of (cloned) DNPU surrogate models when the linear layer that maps formant features to voltages is retrained for each number of DNPUs.

classification accuracies in Figure 7.3d of the main text, where the LDA is used. For a single DNPU SM ($N = 1$) the classification accuracy for the alternative method is worse than when using the LDA, because in the LDA the best classifying inputs V_1 and V_2 are used for this case. However, with increasing N the alternative method starts to outperform the corresponding LDA-based method, finally increasing the classification accuracy from 90.6% to 92.6%.

References

[1] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H.P. Pernice. The rise of intelligent matter. *Nature*, 594(7863):345–355, 2021. doi: 10.1038/s41586-021-03453-y.

[2] K. P. Zauner. From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter. *Lecture Notes in Computer Science*, 3566:47–55, 2005. doi: 10.1007/11527800_4.

[3] Julian Miller, Simon Hickinbotham, and Martyn Amos. In materio computation using carbon nanotubes. *Natural Computing Series*, pages 33–43, 2018. doi: 10.1007/978-3-319-65826-1_3.

[4] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.C. Ruiz Euler, H. Broersma, P.r A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel. Classification with a disordered dopant-atom network in silicon. *Nature*, 577(7790):341–345, Jan 2020. doi: 10.1038/s41586-019-1901-0.

[5] J. F. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. *Proceedings - NASA/DoD Conference on Evolvable Hardware, EH*, pages 167–176, 2002. doi: 10.1109/EH.2002.1029882.

[6] H-C. Ruiz Euler, M. N. Boon, J. T. Wildeboer, B. van de Ven, T. Chen, H. Broersma, P. A. Bobbert, and W. G. van der Wiel. A deep-learning approach to realizing functionality in nanoelectronic devices. *Nature Nanotechnology*, 15 (12):992–998, 2020. doi: 10.1038/s41565-020-00779-y.

[7] M. N. Boon, H.C. Ruiz-Euler, T. Chen, B. van de Ven, U. Alegre Ibarra, P. A. Bobbert, and W. G. van der Wiel. Gradient descent in materio. *arXiv*, 2021. doi: 10.48550/arXiv.2105.11233.

[8] J. F. Miller, S. L. Harding, and G. Tufte. Evolution-in-materio: Evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67, 2014. doi: 10.1007/s12065-014-0106-6.

[9] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks – with an Erratum note. page 223, 2010. doi: 10.1054/nepr.2001.0035.

[10] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002. doi: 10.1162/089976602760407955.

[11] J. J. Steil. Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity. *IEEE International Conference on Neural Networks - Conference Proceedings*, 2:843–848, 2004. doi: 10.1109/IJCNN.2004.1380039.

[12] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009. doi: 10.1016/j.cosrev.2009.03.005.

[13] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019. doi: 10.1016/j.neunet.2019.03.005.

[14] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Evolving carbon nanotube reservoir computers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9726:49–61, 2016. doi: 10.1007/978-3-319-41312-9_5.

[15] Y. Usami, B. van de Ven, D. G. Mathew, T. Chen, T. Kotooka, Y. Kawashima, Y. Tanaka, Y. Otsuka, H. Ohoyama, H. Tamukoh, H. Tanaka, W. G. van der Wiel, and T. Matsumoto. In-Materio Reservoir Computing in a Sulfonated Polyaniline Network. 2102688:1–9, 2021. doi: 10.1002/adma.202102688.

[16] J. Wang, S. Lu, S. Wang, and Y. Zhang. A review on extreme learning machines. 2021. doi: 10.1007/s11042-021-11007-7.

[17] S. Ortín, M. C. Soriano, L. Pesquera, D. Brunner, D. San-Martín, I. Fischer, C. R. Mirasso, and J. M. Gutiérrez. A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron. *Scientific Reports*, 5(1):14945, Oct 2015. doi: 10.1038/srep14945.

[18] M. Dale, S. Stepney, J. F. Miller, and M. Trefzer. Reservoir computing in materio: An evaluation of configuration through evolution. *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, 2017. doi: 10.1109/SSCI.2016.7850170.

[19] H.C. Ruiz-Euler, U. Alegre-Ibarra, B. van de Ven, H. Broersma, P. A. Bobbert, and W. G. van der Wiel. Dopant network processing units: towards efficient neural network emulators with high-capacity nanoelectronic nodes. *Neuromorphic Computing and Engineering*, 1(2):024002, 2021. doi: 10.1088/2634-4386/ac1a7f.

[20] J. M. Hillenbrand. Vowel formant frequency classification data. URL `https://homepages.wmich.edu/$\sim$hillenbr/voweldata.html`.

[21] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv*, 2018. doi: 10.48550/arXiv.1712.06567.

[22] J. Tapson and A. van Schaik. Learning the pseudoinverse solution to network weights. *Neural Networks*, 45:94–100, 2013. doi: 10.1016/j.neunet.2013.02.008.

[23] R. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007. doi: 10.1016/j.neunet.2007.04.017.

[24] C. R. Pernet and P. Belin. The role of pitch and timbre in voice gender categorization. *Frontiers in Psychology*, 3(FEB):1–11, 2012. doi: 10.3389/fpsyg.2012.00023.

[25] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y Shi. Scaling for edge inference of deep neural networks a perspective. *Nature Electronics*, 1(4): 216–222, 2018. doi: 10.1038/s41928-018-0059-3.

[26] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646, 2020. doi: 10.1038/s41586-020-1942-4.

[27] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja, J. Liu, C. D. Wright, A. Sebastian, T. J. Kippenberg, W. H. P. Pernice, and H. Bhaskaran. Parallel convolutional processing using an integrated photonic tensor core. *Nature*, 589(7840): 52–58, 2021. doi: 10.1038/s41586-020-03070-1.

[28] M. Romera, P. Talatchian, S. Tsunegi, F. Abreu Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. Ernoult, D. Vodenicarevic, T. Hirtzlin, N. Locatelli, D. Querlioz, and J. Grollier. Vowel recognition with four coupled spin-torque nano-oscillators. *Nature*, 563(7730):230–234, 2018. doi: 10.1038/s41586-018-0632-y.

[29] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon. Deep physical neural networks enabled by a backpropagation algorithm for arbitrary physical systems. *arXiv*, 2021. doi: 10.48550/arXiv.1712.06567.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch : An Imperative Style , High-Performance Deep Learning Library. *NeurIPS*, 2019.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of machine learning research*, 12:2825–2830, 2011. doi: 10.1289/EHP4713.

8

Conclusions and Outlook

In this thesis, we analysed whether DNPUs have an advantage for neuromorphic computing applications. In this final chapter, we reflect on the obtained results, compare them with the requirements for neuromorphic computing and propose future research directions and application areas.

After introducing the DNPU concept, we showed that DNPUs have tuneable nonlinear behaviour attributed to the variable range hopping (VRH) conduction regime. Based on this tuneable nonlinear behaviour, we showed that DNPUs can be tuned to exhibit a variety of input-output relations. When connecting multiple DNPUs together, they can solve more complex benchmark tasks than an individual DNPU. This indicates that creating networks of DNPUs could be a viable approach to creating neuromorphic hardware. In another approach, we combined DNPUs with linear layers. This approach of realising neuromorphic hardware can be combined with memristive or optical technologies for linear operations. We showed that in this neuromorphic network DNPUs can perform multiple linear and nonlinear operations. Hence, DNPUs can help reduce the number of required components.

Although DNPUs can have advantages in neuromorphic applications, some aspects require further research and optimisation. To be a viable technology, DNPUs need to operate at room temperature and be energy efficient. When making a network of DNPUs they need to have gain to allow the signal to propagate through the network. At the same time, to eliminate the Von Neumann bottleneck, non-volatile memory is also important. Finally, for their general applicability, the fabrication yield of DNPUs needs to be increased. Below we suggest possible approaches that can improve these five aspects of DNPUs.

8.1 Room Temperature Operation of DNPUs

For most of the measurements in this thesis, the DNPUs are operated at 77 K. This is inefficient regarding the energy needed for cooling and the required infrastructure. Since DNPUs are mainly advantageous for edge computing, operation at room temperature is crucial. In Chapter 4 we showed that it is possible to operate a DNPU consisting of a silicon substrate with boron dopants at room temperature. The problem with this approach is that the "back-gate" is decoupled via a p-n junction. This results in a leakage current that governs the output current and the noise. Therefore, we propose to use a silicon-on-insulator wafer[1] to reduce the leakage from the back-gate. By combining this with a different substrate-dopant combination, we can increase the ionisation energy, making it easier to obtain VRH at higher temperatures. This combination could allow operation at higher temperatures without substantial leakage.

8.2 Increasing the Energy Efficiency of DNPUs

The main goal of neuromorphic engineering is fast, energy-efficient computation to perform tasks at which the brain excels. In Chapter 4, we initially estimated the energy efficiency of DNPUs to be more than 100 tera-operations per second per watt (TOP s^{-1} W^{-1}). Although competitive, this energy consumption estimate is based on three considerations. First, reconfigurable Boolean logic requires an ANN with ten linear operations. Second, the limiting factor for the bandwidth is determined by the IV-converter (100 MHz). Third, the estimate is based on a measurement of the power consumption when performing Boolean logic.

Since DNPUs can perform the reconfigurable Boolean logic benchmark. We estimate that a DNPU can perform at least ten linear operations. Since reconfigurable Boolean logic also requires nonlinear operations, of which we do not take into account the added computational requirements, this is a conservative estimate. As mentioned in Chapter 4, the bandwidth estimate is based on the assumption that, since the hopping rate is high, the relaxation time of the network is significantly faster than the bandwidth of the *IV* converter. However, this does not account for the required number of hops and the hopping probability. Kinetic Monte Carlo simulations indicate that this will likely decrease the bandwidth to 1 MHz, reducing the energy efficiency to 1 tera-operations per second per watt[2]. It is important to measure the bandwidth limit of DNPUs to verify this.

Regarding the third consideration, we showed that the power consumption is governed by currents that do not directly contribute to the functionality of the DNPU. We observed that the current at the output electrode is in the nA range while the currents at the other electrodes go up to 1 μA, indicating that redundant currents are running through the DNPU. Since the electrodes far away from the output tune the potential

landscape of the DNPU, it is expected that we can increase the energy efficiency by changing some of these electrodes from Ohmic to capacitive coupling. This capacitive coupling can be achieved using a high-quality oxide layer between the metal and the dopant network. The top and side views of a DNPU with capacitively coupled electrodes (dark grey) are illustrated in Figure 8.1.

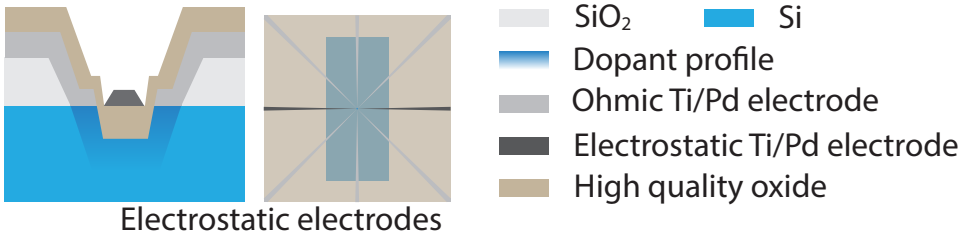


Figure 8.1: Illustration of the side (left) and top (right) of a DNPU where some of the electrodes (dark grey) are electrostatically coupled to the DNPU.

8.3 Achieving Gain in DNPUs

Multiple DNPUs have to be interconnected to create a network capable of solving complex, real-life neuromorphic tasks. In Chapter 6, we took the first step in this direction. In this work we used a digital computer to map the output currents of the DNPU to optimal input voltages of the next DNPU by performing normalisation and re-scaling. In principle, these operations can also be performed using electronic components such as op-amps[3]. However, these consume extra energy, making them a non-ideal approach for realising an interconnected network of DNPUs.

Therefore, it is crucial to try and implement the current to voltage mapping by combining the DNPU and a single resistance. When the input electrodes mainly have an electrostatic influence on the output current, it might be possible to achieve a sensitive enough response in DNPUs. However, when considering the Boolean logic benchmark, for the XNOR task presented in Chapter 4, we observe that a 0.5 V change in the voltage at an input electrode results in a 0.1 nA change in the current at the output electrode. To map this change in current back to a 0.5 V change in voltage we would need a resistance in the order of 5 GΩ, which is higher than the resistance of the DNPU, reducing the output current change.

We thus have to increase the sensitivity of the output current to the input voltages. When performing a task such as X(N)OR, requiring negative differential resistance (NDR), an input voltage changes the output current by adjusting the potential landscape in the DNPU. By placing the input electrode closer to the output electrode we can increase the sensitivity. However, the current from the input may directly flow

to the output. The change in this "direct" current may be larger than the change in current caused by changing the potential landscape, overruling NDR and thus impeding the realisation of X(N)OR functionality. Fortunately, for electrostatically coupled electrodes illustrated in Figure 8.1, it is possible to place the input electrode closer to the output without the direct current problem. Hence, for electrostatically coupled electrodes we can increase the sensitivity, changes in the output current due to changes in the input voltage without overruling NDR. If this sensitivity is large enough, the IV converting resistance could be taken small enough to allow for the propagation of information when interconnecting DNPUs.

8.4 DNPUs and Non-volatile Memory

In this thesis, we have shown that DNPUs can performing analogue neural network-like operations with a reduced number of parameters, as shown in Chapter 6 and 7. However, one of the requirements of neuromorphic hardware is the removal of the Von Neumann bottleneck by performing in-memory computations. By combining DNPUs with linear in-memory computing components, we showed that it is possible to use the linear layer as the non-volatile memory components of the neuromorphic architecture. When using only DNPUs, they need to have non-volatile memory themselves. By building on the electrostatic approach shown in Figure 8.1, it could be possible to use flash memory technology to implement this non-volatile memory by storing charge on electrostatically coupled electrodes[4]. Another method is to use the DNPUs as proposed in Chapter 6, where a linear weight layer tunes the mapping of the task input to all device electrodes (except for the output electrode). In this method, DNPUs only perform non-linear operations and do not require non-volatile memory. The memory should then be present in the weight layer, which could for example, be achieved using a memristive crossbar array[5].

8.5 Improving the Yield of DNPU Fabrication

Increasing the yield of DNPU fabrication results in a more scalable process and makes research on different DNPU designs easier. As mentioned in Chapter 3, the yield is mainly limited by the reproducibility of the dopant concentration at the silicon surface after the reactive ion etching (RIE). We can improve this fabrication step in a few ways. First, using heavier dopants will require less RIE since the desired concentration is closer to the surface. Another approach is to create a wider range in which the desired concentration is reached. We can achieve this by using multiple implantations with different implantation energies and concentrations to create a high concentration at the surface and a platform at the desired concentration. This is illustrated in Figure 8.2 by an implantation simulation[6] using and implantation with 5 keV and $1 \cdot 10^{14}$ ions/cm^2 (blue) combined with an implantation with 20 keV and $2 \cdot 10^{12}$ ions/cm^2 (orange). Here, the green line is the result of the two implantation steps, having a plateau at the operating concentration.

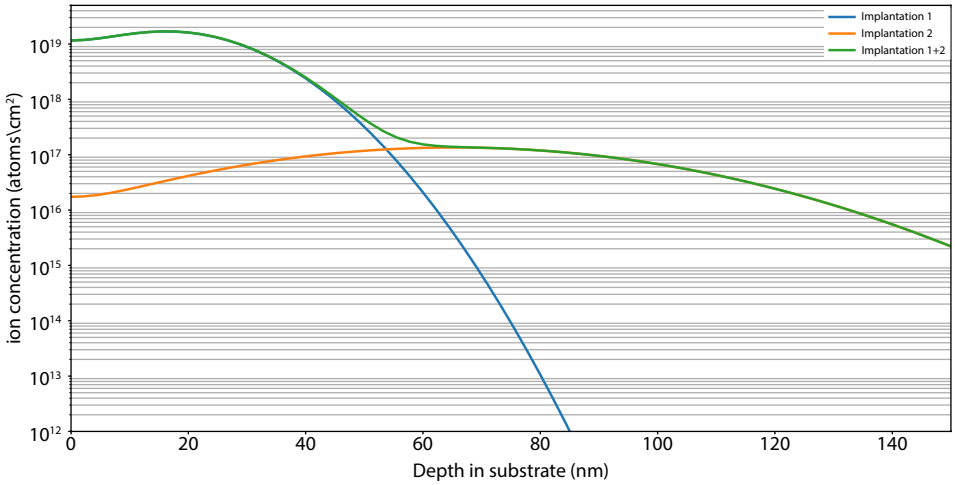


Figure 8.2: Simulation of the ion concentration profile when performing two ion implantation steps. Implantation step 1 (blue) is performed using 5 keV and $1 \cdot 10^{14}$ ions/cm^2. Implantation step 2 (orange) is performed using 20 keV and $2 \cdot 10^{12}$ ions/cm^2. The green line shows the implantation profile when performing implantation steps 1 and 2 on the same wafer [6].

When incorporating DNPUs in neuromorphic hardware, combining the DNPUs with linear components is expected to be the next step. Since the other components can perform the tasks of non-volatile memory and gain, the focus in DNPU research should be on room temperature operation and energy efficiency. When creating a neuromorphic system with combined technologies the signals need to be compatible. Assuming this is the case, the viability of technologies using DNPUs will mainly depend on the bandwidth. When the bandwidth of DNPUs is limiting the energy efficiency of the total system, implementing DNPUs in networks of combined technologies will be inefficient. In this case, the bandwidth of the DNPUs need to be improved, which could be realized using smaller networks or higher temperatures. A smaller network has fewer dopants and thus requires fewer hops to stabilise. At higher temperatures, the hopping rate will increase due to the presence of extra phonons. In short, we have shown in this thesis the advantage of using the nonlinearity of DNPUs in neuromorphic hardware.

References

[1] J.P. Colinge. *Silicon-on-insulator technology: materials to VLSI: materials to Vlsi*. Springer Science & Business Media, 2004.

[2] H. Tertilt, J. Bakker, M. Becker, B. Wilde, I. Klanberg, B. Geurts, W. G. van der Wiel, A. Heuer, and P. A. Bobbert. Mechanism for reconfigurable logic in disordered dopant networks. 07 2021. doi: 10.21203/rs.3.rs-757616/v1.

[3] A. van de Brink. Interconnecting dopant networks. Master's thesis, Master thesis at the University of Twente, September 2021.

[4] P. Pavan, R. Bez, P. Olivo, and E. Zanoni. Flash memory cells-an overview. *Proceedings of the IEEE*, 85(8):1248–1271, 1997. doi: 10.1109/5.622505.

[5] Qiangfei Xia and J. Joshua Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18(4):309–323, Apr 2019.

[6] Brigham Young University. Diffused ion implantation profile calculator and graph, 2022. URL `https://cleanroom.byu.edu/implantcal`. Last accessed 24 January 2022.

Summary

The rise of artificial intelligence (AI) is driven by the increase in processing power of our digital computers and the availability of large amounts of data. This allows a subset of AI that is defined using digital computers, also known as artificial neural networks (ANNs), to become present in today's world. However, the growth of these ANNs is limited by both the Von Neumann bottleneck and the impending end of Moore's law. The Von Neumann architecture separates memory and processing. Since ANNs require many memory accesses, communication between the memory and processor limits the size of ANNs. Simultaneously, Moore's law states that the number of transistors in an integrated circuit doubles every two years. However, we are approaching transistors that need to consist of only a few atoms. This combination results in the requirement for unconventional computing approaches. Neuromorphic hardware, an example of unconventional computing, is inspired by the efficiency of the brain and uses interconnected computing units (similar to neurons in the brain) for AI applications. Neuromorphic engineering focuses on developing computational elements that can be trained and interconnected to perform brain-inspired computation tasks. In this work, we analyse the capabilities of dopant network processing units (DNPUs) in neuromorphic hardware.

As described in Chapter 2, we introduce the three main requirements for neuromorphic systems. First, the system needs to be responsive to external stimuli, such that information is processed. Second, this system needs to achieve the desired response from external stimuli, they need to be trainable. Third, learned states need to be stored in non-volatile memory. We explain the concept of evolution-in-materio (EIM), where a disordered system is trained using a genetic algorithm to perform the desired functionality. Finally, we explain variable range hopping (VRH) and indicate how VRH combined with Coulomb blockade allows DNPUs to be compatible with EIM.

The important aspects of the fabrication and electronic characterisation of DNPUs are described in Chapter 3. The fabrication part is divided into 4-inch, wafer-scale and 1x1 cm^2, chip-scale. On the wafer-scale, masking SiO_2 is used to define the dopant implantation region. These dopants (boron or arsenic) are implanted using ion beam implantation. After implantation, we dice the wafer in 45 1x1 cm^2 chips. These chips are processed using electron beam lithography to define the electrode positions. These electrodes are fabricated using electron beam evaporation and lift-off and are

made of Ti/Pd for boron-doped or Al for arsenic-doped DNPUs. In the final step, we use reactive ion etching to reach the correct dopant concentration for VRH at 77 K. For the electrical characterisation, the DNPUs are placed in liquid nitrogen using a dipstick. This dipstick houses the electronic cables that connect the DNPU to the digital-to-analogue converter (DAC for input voltages) and analogue-to-digital converter (ADC for output voltages). The nA output of the device is mapped to a voltage using a low-temperature IV converter. The converter is placed close to the DNPU to reduce noise.

In Chapter 4, we show that, between 70 K and 160 K, the conduction of DNPUs is governed by variable range hopping (VRH). The tuneable nonlinear behaviour from this VRH is exploited to perform the reconfigurable Boolean logic and feature extraction benchmark. Using the input-output behaviour of the 16 extracted 2x2 pixel features, we simulate that the MNIST handwritten digit classification benchmark can be solved with an accuracy of 96% when combining DNPU feature extractors with a linear layer. For the benchmark tasks in this chapter, the DNPUs are trained using a genetic algorithm performing EIM. The trainability of DNPUs for complex nonlinear tasks makes them interesting for neuromorphic engineering. By applying a back-gate to suppress the conduction from band to VRH, we show that it is possible to perform the reconfigurable Boolean logic benchmark at room temperature.

The disadvantage of using a genetic algorithm to train DNPUs is the large number of measurements involved. When training many benchmarks, a GA becomes time intensive. To find a faster way of training multiple or complex tasks, in Chapter 5, we show that it is possible to train an artificial neural network (ANN) model to behave similarly to the DNPU. This ANN model is trained on the input-output relation of a DNPUs 7-dimensional input space. Using this trained ANN model, also known as a surrogate model (SM), we can use standard deep learning approaches to train the networks. At the initial cost of measuring all the input-output data, subsequent training runs are faster. In Chapter 5, using the SM to train the DNPUs, we show that DNPUs can perform the ring classification benchmark. This ring classification task consists of two sets of concentric rings (classes) separated by a certain distance from one another.

In Chapter 6, we use an SM and Vapnik-Chervonenkis (VC) dimension benchmark to analyse the computational capabilities of DNPUs. VC dimension defines the complexity of a network as the number of binary output tasks this network can perform. We show that a physical DNPU has similar capabilities as a 2-hidden-node ANN. The computation capabilities of the SM are in between an ANN with 2 and 3 hidden nodes. We compare the number of parameters needed to show that, in terms of memory usage, DNPUs have an advantage over fully connected ANNs. Following this analysis, we use the ring classification benchmark task to analyse the added computation capabilities of a network of interconnected DNPUs. Using SMs to train the DNPUs, we attempt to perform the ring classification task for a small separation.

We compare its classification accuracy to that of a network of 5 DNPUs connected in a 2-2-1 configuration. We observe that the network of 5 DNPUs can reach a higher accuracy on this ring classification task. Finally, by using the SM to create a network of multiple SMs, we show that it is theoretically possible to solve the MNIST handwritten digit classification task using a linear input layer and 10 DNPUs. An accuracy of 95 % is achieved on the test data.

In Chapter 7, we move past the standard 1-output DNPU configuration. Utilising multiple output electrodes, we attempt to extract more non-linear computation from the DNPU. This approach is inspired by the field of extreme learning machines (ELM) and reservoir computing (RC) where the network creates a complex mapping from the input to its internal states. These internal states are linearly mapped to the desired output to perform the benchmark task. In this Chapter, we show that the ELM framework allows us to extract the non-linear behaviour from a single 12 electrode DNPU. We build on this concept by showing that DNPUs require a certain degree of control to create a potential landscape capable of extracting more varied nonlinear behaviour. This extra variation allows us to move from solving VC-dimension 5 (5 inputs) to VC-dimension 7 (7 inputs). This increase shows that we can solve 96 extra binary classification tasks. Operating the DNPU in the tuneable ELM mode, using the nonlinearity of 3 DNPUs in parallel, the formant-based vowel recognition benchmark can be solved with an accuracy of 89.9 %. Furthermore, by comparing DNPUs to ANNs the number of parameters/ memory accesses needed is reduced from 72 to 6 and scales better for DNPUs.

The results in this thesis show both the potential and limitations of using DNPUs for neuromorphic computing. The highly tuneable nonlinear response combined with global tuneability allows DNPUs to perform complex tasks that require multiple operations in standard AI implementations. By combining the non-linearity of DNPUs with the memory of linear implementations for neuromorphic engineering, it seems likely that networks combining different material platforms can be created that efficiently perform brain-inspired computation. Using DNPUs, we show the advantages of using the tuneable nonlinear behaviour of disordered networks for neuromorphic hardware.

Samenvatting

De opkomst van kunstmatige intelligentie (KI) wordt gedreven door de groei in rekenkracht van onze digitale computers en de beschikbaarheid van grote hoeveelheden data. Hierdoor zijn kunstmatige neurale netwerken (KNN) een subgroep van KI, die gebruikt worden in ons dagelijkse leven. Maar, de groei van KNN's wordt beperkt door het Von Neumann knelpunt en het naderende einde van de wet van Moore. De Von Neumann-architectuur scheidt geheugen en processor. Omdat een KNN vaak informatie uit het computergeheugen nodig heeft, limiteert de communicatie tussen de processor en het geheugen de grote van een KNN. De wet van Moore stelt dat het aantal transistoren in een Geïntegreerde schakeling elke twee jaar verdubbelt. Maar, we komen steeds dichterbij een transistor die uit een paar atomen zou moeten bestaan. Deze combinatie zorgt voor de vraag naar onconventionele computers chips. Neuromorphische hardware, een voorbeeld van zo'n onconventionele chip, wordt geïnspireerd door de efficiëntie van het brein en maakt gebruik van onderlinge verbindingen tussen individuele rekeneenheden (vergelijkbaar met de neuronen in het brein) om KI te implementeren in hardware. Het ontwerpen van neuromorphische computers is gericht op het ontwikkelen van zulke rekeneenheden die, door ze aan elkaar te koppelen, taken waar het brein goed in is efficiënt uit kunnen voeren. In dit werk kijken we naar de potentie van wanordelijke doterings netwerken (DNPUs) voor neuromorphische computers.

Zoals beschreven in hoofdstuk 2, introduceren we de drie belangrijkste vereisten voor neuromorphische systemen. Ten eerste moet het systeem reageren op externe signalen zodat het systeem deze informatie kan gebruikt om de gewenste berekeningen uit te voeren. Ten tweede moet het mogelijk zin om het systeem te beïnvloeden om het gewenste gedrag te vertonen, het moet te trainen zijn. Ten derde moet het systeem in staat zijn dit geleerde gedrag op te slaan in niet-vluchtig geheugen. In het hoofdstuk introduceren we het concept van evolutie-in-materie (EIM). Met EIM is het mogelijk een wanordelijk system te trained door gebruik te maken van een genetisch algoritme (GA). Ten slotte beschrijven we hoe hopping in combinatie met Coulomb blokkade ervoor zorgt dat DNPU's gebruikt kunnen worden voor EIM.

De belangrijkste aspecten van de fabricage en elektronische karakterisatie van DNPU's worden beschreven in hoofdstuk 3. De beschrijving van de fabricage is opgesplitst in de 4-inch waferschaal processen en de 1x1 cm^2 chip processen. Op de waferschaal

gebruiken we SiO_2 om het implantatie gebied te definiëren. In het gebied met minder SiO_2 doteren we de DNPU, met borium of arseen, door ionen straal implantator te gebruiken. Na de implantatie wordt de wafer in 1x1 cm^2 chips gezaagd. Op deze chips worden de posities van de elektroden gedefinieerd door elektronen bundel lithographie. De elektroden worden gefabriceerd door metaal te verdampen met een elektronen straal waarna het metaal een laag vormt op de chips. Hierna gebruiken we lift-off om de Ti/Pd (voor boor gedoteerd Si) of de Al (voor arseen gedoteerd Si) elektroden te vormen. In de laatste stap etsen we het gedoteerde silicium met reactieve ionen om de juiste doterings concentratie voor hopping bij 77 K te bereiken. Voor de elektrische karakterisering plaatsen we de DNPU's in vloeibare stikstof met behulp van een dipstick. Deze dipstick bevat de elektronische kabels die de DNPU verbindt met de digitaal-naar-analoog-omzetter (voor ingangsspanningen) en de analoog-naar-digitaal-omzetter (voor uitgangsspanningen). De nA stroom van de DNPU wordt omgezet naar een spanning met behulp van een IV-converter. De converter is dicht bij de DNPU geplaatst om ruis te verminderen.

In hoofdstuk 4 laten we zien dat, tussen 70 K en 160 K, hopping (VRH) conductie plaats vindt door de DNPU's. Het trainbare, niet-lineaire gedrag gelinkt aan VRH wordt benut om herconfigureerbare Booleaanse logica en de kenmerk extractie taken uit te voeren. Met behulp van het in-uit gedrag van de 16 2x2 pixel kenmerken, simuleren we dat de MNIST handgeschreven cijferclassificatie taak kan worden opgelost met een nauwkeurigheid van 96% door het gedrag van de DNPU kenmerk extractie taak te combineren met een lineaire laag. Voor de taken die zijn uigevoerd in dit hoofdstuk, trainen we de DNPU's met behulp van een genetisch algoritme dat EIM uitvoert. De trainbaarheid van DNPU's voor complexe niet-lineaire taken maakt ze interessant voor neuromorfische toepassingen. Door een spanning op het substraat aan te brengen kunnen we de band geleiding onderdrukken. In dit geval vindt voornamelijk VRH conductie plaats. Hiermee laten wezien dat het mogelijk is om de herconfigureerbare Booleaanse logica taak uit te voeren op kamertemperatuur.

Het nadeel van het gebruik van een genetisch algoritme (GA) om DNPU's te trainen is het grote aantal metingen dat nodig is om het gewenste gedrag te vinden. Bij het trainen van veel verschillende taken wordt een GA tijdrovend. Een snellere manier om DNPUs te trainen laten we in Hoofdstuk 5 zien. Dit doen we door een model van een KNN te trainen om zich op dezelfde manier te gedragen als de DNPU. Dit KNN-model is getraind op het ingangs-uitgangs gedrag van de 7-dimensionale input dimensie van DNPU's. Met behulp van dit getrainde KNN-model, ook bekend als een surrogaatmodel (SM), kunnen we standaard deep learning technieken gebruiken om de DNPUs te trainen. Tegen de initiële kosten van het meten van al het ingangs-uitgangs gedrag, kunnen alle daaropvolgende trainingen sneller worden uitgevoerd. In hoofdstuk 5, waarbij we het SM gebruiken om de DNPU's te trainen, laten we zien dat DNPU's de ringclassificatie taak kunnen uitvoeren. Deze ringclassificatie taak bestaat uit twee concentrische ringen (klassen) die met een bepaalde afstand van elkaar zijn gescheiden.

In Hoofdstuk 6 gebruiken we een SM en de Vapnik-Chervonenkis (VC) dimensie taak om de rekencapaciteiten van DNPU's te analyseren. VC-dimensie definieert de complexiteit van een netwerk als het aantal binair uitvoerbare taken dat het netwerk kan leren. We laten zien dat een fysieke DNPU een vergelijkbare rekencapaciteit heeft als een KNN met 2 twee nodes in de verbogen laag. De rekencapaciteit van het SM zit tussen dat van een KNN met 2 en 3 in de verbogen laag. We laten zien dat het aantal parameters dat een DNPU nodig heeft voor een beaaplde taak lager is dan bij een volledig verbonden KNN. Na deze analyse gebruiken we de ringclassificatie taak om de toegevoegde rekencapaciteit van een netwerk van onderling verbonden DNPU's te analyseren. Door SM's te gebruiken om DNPU's te trainen laten we zien dat een enkele DNPU niet goed is in de ringclassificatie taak wanneer scheiding tussen de twee datasets klein is. We vergelijken de uitslag van deze enkele DNPU met een netwerk van 5 DNPU's die zijn aangesloten in een 2-2-1-configuratie. Hier zien we dat het netwerk van 5 DNPU's in staat is om een hogere nauwkeurigheid te bereiken voor de ringclassificatie taak. Ten slotte, met behulp van SM's, laten we zien dat het theoretisch mogelijk is om de MNIST handgeschreven cijferclassificatie taak op te lossen met behulp van een lineaire invoerlaag en 10 DNPU's. Op de testdata wordt een nauwkeurigheid van 95 % behaald. Hiermee is de theoretische hoeveelheid DNPU's nodig voor deze taak flink verlaagd.

In Hoofdstuk 7 bouwen wij verder op de standaard gebruikte 1-uitgang DNPU-configuratie. Door gebruik te maken van meerdere uitgangselektroden, proberen we meer niet-lineaire gedrag uit de DNPU te halen. Deze benadering is geïnspireerd door extreme learning machines (ELM) en reservoir computing (RC), waarbij het netwerk een complexe mapping creëert van de informatie in de ingangs elektrode naar zijn interne toestanden. Op Deze interne toestanden wordt een lineaire operatie toegepast die de staten naar het gewenste gedrag transfomeerd. De waarde van de lineaire elemented worden getrain om de juiste taak uit te voeren. In dit hoofdstuk laten we zien dat het gebruik van de ELM technieken ons in staat stelt meer niet-lineair gedrag uit een enkele DNPU met 12 elektroden te halen. We bouwen voort op dit concept door te laten zien dat DNPU's een zekere mate van externe invloed vereisen om een staat te creëren die meer gevarieerd niet-lineair gedrag vertoont. Deze extra variatie zorgt ervoor dat we nu in plaats van VC-dimensie 5 (5 datapunten), VC-dimensie 7 (7 datapunten) in onze rekenkracht taak kunnen bereiken. Deze toename geeft aan dat we nu 96 extra binaire classificatietaken kunnen oplossen. Door de DNPU in de trainbare ELM-modus te gebruiken, met behulp van de niet-lineariteit van 3 DNPU's in parallel, kan een klinkerherkennings taak worden opgelost met een nauwkeurigheid van 89,9 %. Bovendien, door DNPU's te vergelijken met KNN's, wordt het aantal benodigde parameters/geheugentoegangen verminderd van 72 naar 6. Hiernaast schaalt de groei van de hoeveelheid benodigde parameters beter bij het gebruik van DNPU's.

De resultaten in dit proefschrift laten zowel de potentie als de beperkingen van DNPU's voor neuromorfisch computergebruik zien. Het trainbare niet-lineaire gedrag in combinatie met de globale elektrostatische invloed stelt DNPU's in staat om complexe taken uit te voeren die meerdere operaties vereisen in standaard KI-implementaties. Door de niet-lineariteit van DNPU's te combineren met het geheugen van lineaire implementaties voor neuromorfische chips lijkt het mogelijk om netwerken te criëren die verschillende materialen combineert. Op deze manier ie het theoretisch mogelijk dat hersengeïnspireerde berekeningen efficiënt uitgevoerd kunnen worden. Met DNPU's als een voorbeeld, laten we zien dat het gebruik van het trainbaar niet-lineaire gedrag van ongeordende netwerken voordelen kan hebben voor neuromorfische toepassingen.

List of Publications

Publications used for this Thesis

1. T. Chen, J. van Gelder, **B. van de Ven**, S. V. Amitonov, B. de Wilde, H.-C. Ruiz-Euler, H. Broersma, P. A. Bobbert, F. A. Zwanenburg, W.G. van der Wiel, Classification with a disordered dopant-atom network in silicon. Nature. **577**, 341–345 (2020). https://doi.org/10.1038/s41586-019-1901-0

2. H.-C. Ruiz-Euler, M. N. Boon, J. T. Wildeboer, **B. van de Ven**. T. Chen, H. Broersma, P. A. Bobbert, W. G. van der Wiel, A deep-learning approach to realizing functionality in nanoelectronic devices. Nat. Nanotechnol. **15**, 992–998 (2020). https://doi.org/10.1038/s41565-020-00779-y

3. H.-C. Ruiz-Euler, U. Alegre-Ibarra, **B. van de Ven**, H. Broersma, P. A. Bobbert, W. G. van der Wiel, Dopant network processing units: towards efficient neural network emulators with high-capacity nanoelectronic nodes. Neuromorph. Comput. Eng. **1** 024002 (2021). https://doi.org/10.1088/2634-4386/ac1a7f

4. **B. van de Ven**, U. Alegre-Ibarra, P.J. Lemieszczuk, P. A. Bobbert, H.-C. Ruiz-Euler, W. G. an der Wiel, Dopant network processing units as tuneable Extreme Learning Machines. In Preparation.

Additional Publications

– A. J. Sousa de Almeida, A. Márquez Seco, T. van den Berg, **B. van de Ven**, F. Bruijnes, S. V. Amitonov, F. A. Zwanenburg, Ambipolar charge sensing of few-charge quantum dots. Phys. Rev. B. **101**. 201301 (2020). https://doi.org/10.1103/PhysRevB.101.201301

– M. N. Boon, H.-C. Ruiz-Euler, T. Chen, **B. van de Ven**, U. Alegre-Ibarra, P. A. Bobbert, W. G. van der Wiel, Gradient Descent in Materio. arXiv:2105.11233 (2021)

– Y. Usami, **B. van de Ven**, D. G. Mathew, T. Chen, T. Kotooka, Y. Kawashima, Y. Tanaka, Y. Otsuka, H. Ohoyama, H. Tamukoh, H. Tanaka, W. G. van der, T. Matsumoto, In-Materio Reservoir Computing in a Sulfonated Polyaniline Network. Adv. Mater. **33**, 2102688. 2021, https://doi.org/10.1002/adma.202102688

Acknowledgements

To me, the value of doing a PhD does not only come from my growth as a scientist but more importantly from my growth as a person. However, achieving growth is rarely achieved alone. Therefore, in this chapter, I want to thank the people that help me not only complete my PhD but also grow as a person in the process. Since there are so many people that have contributed I apologise in advance if I happen to forget you.

The PhD journey can not start without a position and the corresponding supervisors. **Wilfred** and **Peter**, thank you for having me in the NanoElectronics group. **Wilfred**, you always gave me the freedom to pursue my ideas while being ready to ask critical questions when needed. Besides this, you helped me learn to set my limits while keeping me focused the end goals. **Peter**, thank you for being ready to play the devil's advocate keeping us sharp and critical. Your help in clarifying theoretical concepts and most importantly in helping polish my ideas such that they are also clear for others was crucial for this me and this thesis.

Floris, thank you for introducing me to the NanoElectronics (NE) group during my bachelor and master thesis and well as always being ready to lend advice when my worries were getting the better of me. **Alejandro**, thank you for the supervision of my master thesis as well as for helping with a smooth transition from a master student to a PhD candidate within NE. **Tao**, your supervision, guidance and in-depth discussions were crucial for making this thesis what it is today.

Besides Tao, I first want to thank everyone else from the Brains part of NE. Without **Hans-Christian** I would still be lost in the world of ANNs and programming. **Unai**, the debugging sessions were crucial. The great times spend with you and **Rocio** made my PhD an enjoyable experience. **Reza**, being able to discuss as well as vent to one another in this last year was crucial to keep me sane. **Marc** and **Lorenzo**, the activities and vrimibo's this last half a year would not have been the same without you there. **Reinier**, my latest office mate. It was nice sharing an office with you for however short it was, enjoy your new seat at the window.

But of course, the rest of NE was just as, if not more, important during the PhD period. **Guus**, thank you for being the mirror I sometimes need and being a great friend whom I can always talk to. NanoElectronics would not be the same without **Dilu**. Thank you for always being there to provide advice, social interaction and be the bridge between different people in the group. **Pavel**, always in for a laugh and my procrastination partner in crime. The don't starve and Fifa sessions were always a lot of fun. You are next in line good luck, you got this. **Robert**, **Michal**, **Sander** and **Yigitcan**, you guys made the beginning of my PhD a great experience and I will never forget the trip to Izmir. **Amber**, it was a pleasure getting to know you. And remember, you do not need coffee to finish a PhD thesis.

A research group would fall apart without its secretary. **Judith** and **Karen**, thank you for all the little things you did behind the scenes. **Michel**, from the start your knowledge of a wide variety of subjects has amazed me. Keep sharing your knowledge since it is a driving force in NE. **Johnny**, without your advice and help with fabrication, this thesis would never have existed. **Martin**, thank you for your time and patience in designing new measurement setups as well as the help in supervising students when their projects were too far away from my expertise. **Antonio**, thank you for the tennis, good food and help in the lab. We are still waiting for a short dance workshop.

During my PhD I got to work with a few PhD from other universities each of which showed me different approaches to research and views on life. **Yuki**, The work on SPAN networks was interesting and the published paper shows the power of the collaborations between different universities. **Hadi**, even though short, our time looking at short-long term memory was very educational. **Qiao**, It was a pleasure to work with you on your disordered networks of nanowires with silver nanoparticles. **Fabiana**, getting to know and work with you was a pleasure. I will remember this period fondly, especially the two weeks of goodbye celebrations.

During a PhD you are not only learning, you are also supervising students. Here I want to thank the master students for the experience and knowledge I gained from supervising them. **Michelangelo**, you came from Italy to do your internship in our group by helping us develop a new setup allowing use to better characterise DNPUs. **Jochem**, you where very independent and ambitious. By designing your own project you showed a lot of independence. At the same time, your work contributing to Ch 5 shows your willingness to collaborate on projects as well. The biggest challenge for me was to keep your ambitions in check to reach a feasible project. **Pawel**, Due to circumstances your thesis was not always easy. Nevertheless, your contributions to the work presented in Ch 7 of this thesis are highly appreciated. **Aernout**, thank you for choosing to do both your master and bachelor thesis with me as your (co-)supervisor and for helping with the development of the setups crucial to perform the measurements in this thesis. **Engbert**, the last master student I supervised. It was a pleasure working with you and thank you for all the extra work you did for NE when

I was busy writing this thesis.

Besides master students, I of course also supervised bachelor students. **Tuomo**, **Rik**, **Jardi** and **Lars**, it was a pleasure working with you. I hope you could learn something from me and are doing well during your next steps in life.

A thesis cannot be defended without a committee. Therefore, I want to thank **Hans**, **Tamalika**, **Susan** and **Hajo**, for being part of this committee, for the valuable feedback and kind words regardign this thesis. I specifically want to thank **Hajo**, for indirectly being there every step of the way, to always brighten the mood and enjoy the NEvents.

I want to thank some of the people from **POF**, the nice gathering and diners were always fun. Thank you for all the great times in the sand playing beach volleyball.

David, **Roan**, **Rinder**, **Nils** and **Jan**, it is amazing how after all this time we are still in contact. The LAN party's playing some AOE2 and Wolfenstein ET are always a joy. We should quickly organise another one.

Koen, I want to thank you for always being there, whether it is to be my paranymph, help us move or just for a nice evening of gaming. **Yorick**, from bachelor to master and now even a Phd doing this at the same time and learning from one another has always been a pleasure. Good luck with the last few steps. **Shu**, thank you for always being someone I can "discuss" different aspects of our PhD life with.

Fanny and **Piet**, thank you for always being there supporting and believing in me. Without your support to try and fulfill my potential, I would never have finished this thesis or even stood at the starting line of the journey. **John**, **Claudia**, **Rik**, **Devin** and **Devon**, thank you all for always creating an environment where we can go to have a good time and forget about all the challenges surrounding the PhD such that I could relax. Without it, I would not have made it. **Hannie**, **Willem** and **Elise**, thank you for being there throughout my PhD journey.

Maaike I can not imagine having gone through this journey without you. The continuous sharing of new ideas, views and concepts allowed us to grow into even better versions of ourselves. I am certain that during all our experiences and adventures we will keep challenging and supporting one another and keep on growing.