Design of Fully Analogue Artificial Neural Network with Learning Based on Backpropagation

Filip PAULU, Jiri HOSPODKA

Dept. of Circuit Theory, Czech Technical University in Prague, Technická 2, 160 00 Prague, Czech Republic

{paulufil, hospodka}@fel.cvut.cz

Submitted July 7, 2020 / Accepted April 7, 2021

Abstract. *A fully analogue implementation of training algorithms would speed up the training of artificial neural networks. A common choice for training the feedforward networks is the backpropagation with stochastic gradient descent. However, the circuit design that would enable its analogue implementation is still an open problem. This paper proposes a fully analogue training circuit block concept based on the backpropagation for neural networks without clock control. Capacitors are used as memory elements for the presented example. The XOR problem is used as an example for concept-level system validation.*

Keywords

Fully analogue, analogue circuit, neural network, neuromorphic, backpropagation

1. Introduction

The edge computation architectures minimise the movement of the data by performing the computations close to the sensors. The significant computing power in this domain comes with the motivation to design hardware for near-sensor data analysis, which is not affected by the von Neu-mann bottleneck [1–4]. This field has also been influenced by the recent advancement of the Internet-of-Things (IoT) devices [5–7].

Neuromorphic computing draws inspiration from Artificial Neural Networks (ANNs), where learning is done in analogue or mixed-signal domain [1,3,8,9]. Training of these networks requires a learning algorithm such as backpropagation using gradient descent [7,10–13] to implement hardware for real-time training [2, 7, 14, 15]. Despite the widespread use of digital backpropagation for the training of ANNs, its analogue implementation is still an open problem [7], [10].

While hardware-based ANNs currently exist, their training has to be performed with specialised software and FPGA-based units. An analogue implementation of the backpropagation would enable the creation of fully hardware-based ANNs, which could be trained on-chip [2], [15]. Even though this can diminish structural flexibility, it also offers a significant speed-up in the training process of artificial neural networks [7, 10, 11].

For that, it is necessary to use an electronic component that can work as analogue memory and is possible to update. For this, state of the art articles are using memristors or memristive crossbar arrays [2–4, 6, 7, 10, 16]. Unlike CMOS technology, memristors cannot yet work precisely for large-scale multi-level simulations [7], [10]. That is why in this proposal of block concept, capacitors are used as memory elements. Capacitors in VLSI occupy a lot of space, but for fast near-sensor application it is not always necessary to use large neural network structures. The capacitors can be changed to other technologies with transistor-level designing.

A large number of designs of an on-chip learning processes are done in steps [7, 10, 11]. This allows to use more flexible structure, but it slows down the learning process and disables the usage of the analogue input from sensors without sampling. That is why this design does not use any clock control. It avoids synchronization signals, and it makes this concept fully-analogue and fully-parallel [2], [15].

For above-mentioned reasons, this article presents and verifies the novel concept of a fully analogue training process of an artificial neural network circuit implementation inspired by the backpropagation and based on the gradient descent.

2. Background

The speedup of the training process of a neural network is based on several principles in this article. The training process is designed to be fully parallel [2, 7, 15]. Further, any clock control that holds back the propagation signal is avoided. The training process is realized by analogue electric circuit feedback; thus, it is fully analogue [15]. The design consists of two parts: forward propagation and backward propagation. Inputs, outputs and weights of a classical neural network [12] are represented by: input x_i as V_{in_i}, output $\hat{y}_i$ as V_{out_i}, target y_i as V_{out_i} and weight w_i as V_{w_i}. For the transistor-level implementation, current can be used instead of voltage for x_i, $\hat{y}_i$ and y_i.

The forward propagation of the designed neural network is defined at the level of a single neuron by

$$V_{\text{out}} = S\left(\sum_{i=0}^{N} V_{w_i} V_{\text{in}_i}\right) \tag{1}$$

where V_{out} is the output signal (here, the voltage of this neuron), $S(\cdot)$ is the activation function of this neuron, V_{in_i} is one of the input signals of this neuron and V_{w_i} is the weight for input i, where for $i = 0$ the V_{in_0} is a bias input.

There are two parts to the backpropagation between layers. The first part solves the calculation of an error at the output layer of a neural network (Type_1). It is possible to define an error of the NN's output layer as a Mean Square Error (MSE) [12], [17] by

$$E = \frac{1}{2}\sum_{i=0}^{N}(V_{\text{out}_i} - V_{\text{target}_i})^2 \tag{2}$$

where E [V^2] is the error, N is the number of outputs, V_{target_i} is the ideal output and V_{out_i} is the obtained output signal. The second part solves the backpropagation of the error between the hidden layers of a neural network (Type_2).

The backpropagation circuit implementation is based on a well-known backpropagation algorithm with the stochastic gradient descent method where the update of the weight [12], [13] is defined by

$$w_{k+1} = w_k - \eta\frac{\partial E}{\partial w_k} \tag{3}$$

where w_k is a weight at step $k \in \mathbb{N}$, E is the error and η is a learning rate.

The fully Analogue Artificial Neural Network (AANN) does not operate by separate and distinct steps. The proposed design for the update of the weight uses

$$V_w(t) = V_w(t_0) - \eta\int_{t_0}^{t}\frac{\partial E(\tau)}{\partial V_w(\tau)}\mathrm{d}\tau \tag{4}$$

where $V_w(t)$ is a function of weight at continuous time $t \in \mathbb{R}$.

3. Circuit Design

The proposed AANN is based on a structure that allows the construction of most types of neural networks such as RNN, CNN etc. Each of these structures consists of the same cells, called neurons. The proposed hardware concept of a fully analogue neuron with training is illustrated in Fig. 1. The signals can be realised by voltages or currents depending used technologies. For example, a multiplier can be implemented in both voltage and current mode. However the capacitors are necessary to charge by current and voltages of capacitors are used as a memory quantity, see Sec. 3.2. It consists of voltage multipliers, capacitors representing weights of a neuron, and blocks representing activation function and its derivative. The green boundary indicates the forward propagation part of the circuit. The blue boundary indicates the backpropagation part of the circuit.

The multiplier block has the schematic symbol shown in Fig. 2. It is a function which multiplies two voltages [19], [21] and whose output is a current calculated by

$$I_{\text{out}} = K_m \cdot V_{\text{in}_1} \cdot V_{\text{in}_2} \tag{5}$$

where K_m [A/V^2] is the ratio of output to the product of inputs.

Figure 3 shows the schematic symbol of the block representing an activation function. A sigmoid is the most commonly used activation function, which is described by

$$V_{\text{out}} = \frac{V_{\text{amp}}}{1 + \mathrm{e}^{-I_{\text{in}}/I_{\text{ref}}}} \tag{6}$$

where V_{amp} is a constant determining the maximum possible voltage and I_{ref} is a referential current.

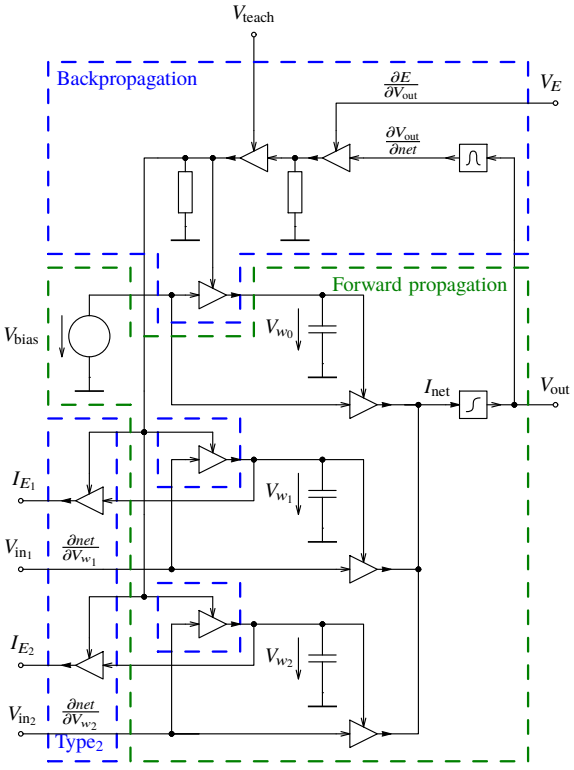

Fig. 1. The block implementation of an analogue neuron with learning and two inputs.

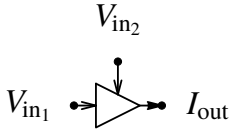

Fig. 2. Symbol of multiplier.

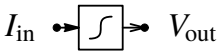

Fig. 3. Symbol of activation function.

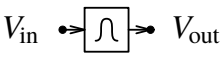

Fig. 4. Symbol of derivation of activation function.

Figure 4 shows the schematic symbol of the block representing the derivative of an activation function. In the sigmoid activation function, the block is described by

$$V_{\text{out}} = K_m \frac{e^{V_{\text{in}}/V_{\text{ref}}}}{(e^{V_{\text{in}}/V_{\text{ref}}} + 1)^2} \tag{7}$$

where K_m is a proportionality voltage constant and V_{ref} is a referential voltage.

3.1 Implementation of Forward Propagation

There have been many solutions to the problem of hardware implementation of forward propagation of neural networks [1,7–10,15,18,19]. The whole behaviour is described by (1), and the analogue design is shown in Fig. 1 in the green area. The weights correspond to the voltages of capacitors [15]. Each input is connected to the multiplier, whose output is a current given by (5). The activation function is applied to the sum of these currents I_{net} [18].

The crucial element in this type of implementation is the capacitor. Each neuron has $N + 1$ capacitors as weights, where N is the number of input. For example, neural network in Fig. 7 has 17 capacitors. However, bigger chip area and higher current consumption are required for bigger structures. The exact implementation depends on the technologies, but there could be tens of thousands of capacitors on a 1 mm^2 of a chip. That is sufficient for a large number of fast near-sensor applications.

3.2 Implementation of Backpropagation

The training process is described by (4). The weight update is implemented by charging a capacitor, which is described by

$$V_w(t) = \frac{1}{C} \int_{t_0}^{t} I(\tau)\mathrm{d}\tau + V_w(t_0). \tag{8}$$

For this implementation, the current is denoted as I_{charged} and defined by

$$I_{\text{charged}} = -K_\eta \frac{\partial E}{\partial V_w} \tag{9}$$

where K_η [S] is the conductance coefficient. Substitution of (9) to (8) gives

$$V_w(t) = V_w(t_0) - \frac{K_\eta}{C} \int_{t_0}^{t} \frac{\partial E(\tau)}{\partial V_w(\tau)}\mathrm{d}\tau \tag{10}$$

where t is the continuous-time and C is the capacitance. Comparison of (10) and (3) leads to the definition of learning rate as $\frac{K_\eta}{C} \equiv \eta$. The process of a weight update is illustrated in Fig. 5.

The partial derivative of the error with respect to a weight V_w is calculated using the chain rule twice

$$\frac{\partial E}{\partial V_w} = \frac{\partial E}{\partial V_{\text{out}}} \frac{\partial V_{\text{out}}}{\partial net} \frac{\partial net}{\partial V_w} \tag{11}$$

where $net = \sum_{i=0}^{N} V_{w_i} V_{\text{in}_i}$ [V^2].

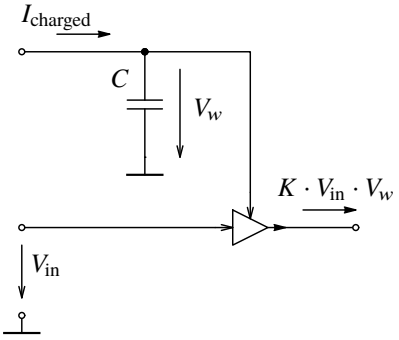

Fig. 5. Analogue implementation of a weight.

The first partial derivative from (11) is denoted in Fig. 1 as voltage V_E:

$$\frac{\partial E}{\partial V_{\text{out}}} = V_E. \tag{12}$$

The last partial derivative from (11) is solved as

$$\frac{\partial net}{\partial V_w} = V_{\text{in}}. \tag{13}$$

The remaining partial derivative cannot be solved generally. However, this is a derivative of the activation function. If the activation function is known, it is possible to calculate and implement this partial derivative as a separate sub-circuit.

The voltage V_{teach} determines if the error is propagated and weights are updated in the neuron. If this voltage is set to 0 V, only the forward-propagation part of the circuit is active. If $V_{\text{teach}} > 0$, the weights are being updated and the network is learning. The magnitude of V_{teach} determines the magnitude of the analogue learning rate. In simulations in Sec. 4, V_{teach} is set to 0 V for the forward-propagation mode and to 1 V for the learning mode.

3.3 Implementation of Backpropagation Between Layers

For simplicity, the whole circuit of the analogue neuron shown in Fig. 1 will be represented by the symbol shown in Fig. 6.

An example of an AANN composed of these neurons is presented in Fig. 7, which demonstrates a neural network with three layers. It contains two input signals represented as voltage sources V_{in_1} and V_{in_2}. The hidden layer in the figure is created by three neurons N_1^2, N_2^2 and N_3^2.

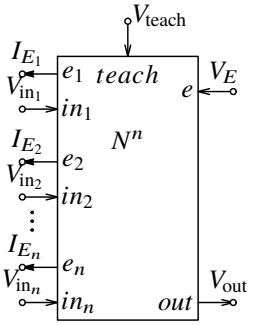

Fig. 6. Symbol of a analogue neuron with n inputs.

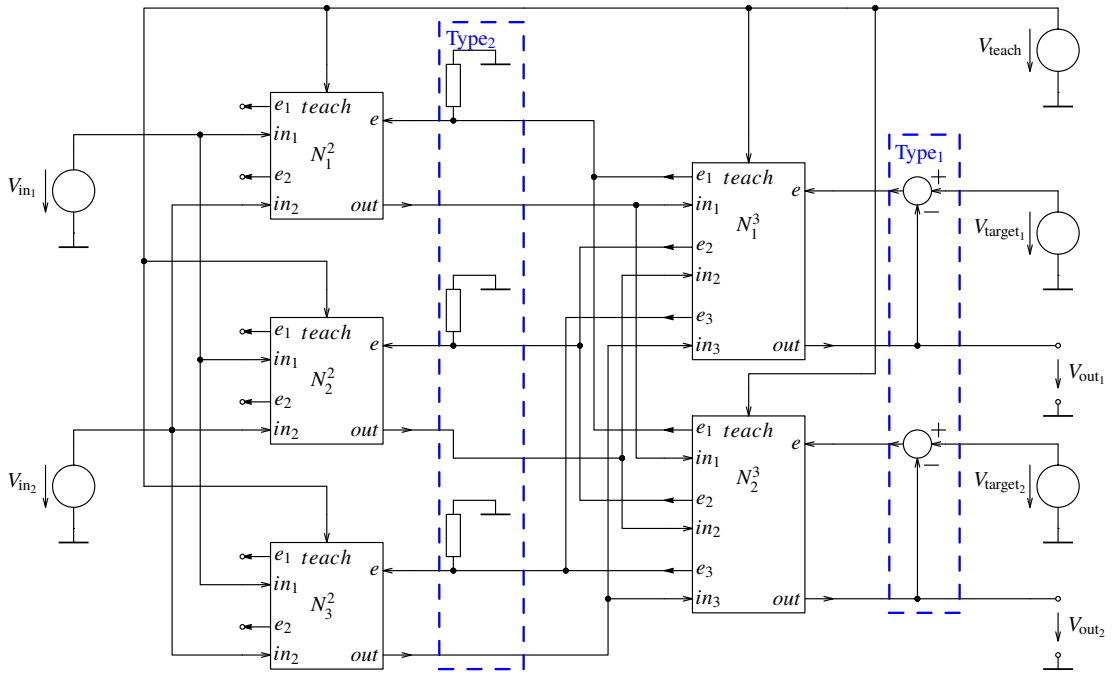

Fig. 7. The block implementation of analogue neural network with two inputs, three neurons in the hidden layer and two outputs.

The output layer is created by neurons N_1^3 and N_2^3. The blue area on the right side (denoted as Type_1) represents the error propagation for the output layer. It comes from the first partial derivative of (11) for the output layer and can be calculated as

$$\frac{\partial E}{\partial V_{\text{out}}} = V_{\text{out}} - V_{\text{target}}. \tag{14}$$

The backpropagation of each neuron in the hidden layers is dependent on the errors of all neurons that indirectly connect it to the output. So, for the hidden layers, the first partial derivative from (11) cannot be calculated directly as in the output layers. This partial derivative is then calculated as

$$\frac{\partial E}{\partial V_{\text{out}}} = \sum_n \frac{\partial E}{\partial V_{\text{out}_n}} \frac{\partial V_{\text{out}_n}}{\partial net_n} \frac{\partial net_n}{\partial V_{\text{out}}} \tag{15}$$

where n is a neuron connected to the output. The sum is again implemented as summation of currents I_{E_n} shown in Fig. 6. It is then converted to voltage V_E as shown in the Type_2 area in Fig. 7.

Implementation of the first two partial derivatives in the sum in (15) is shown in Fig. 1 in the top part of the backpropagation circuit. The last one is solved as

$$\frac{\partial net_n}{\partial V_{\text{out}}} = V_{w_n} \tag{16}$$

which is implemented in Fig. 1 in the Type_2 area.

4. Circuit Simulation

The aim of the simulation is to verify that the learning process of the designed concept of the neural network works. For this purpose, it is not necessary to implement and simulate the circuit on the transistor level. That is the reason why the blocks are defined by mathematical expressions. The whole concept is designed for analogue near-sensor applications. It means that it is not essential to compare the accuracy results with non-analogue neural networks. At the same time, it is not necessary to use complicated datasets such as MNIST for this verification. Spice OPUS is used as an engine to analyze the behaviour and parameters of the resulting AANN structure.

4.1 Learning Process

The first simulation demonstrates the learning process of a designed neural network by transient analysis. The simulated AANN is made up of three parts; one input, two neurons in a hidden layer and one output. The dataset contains two rows. For illustration, after each epoch, the V_{teach} is toggled. The result of this simulation is shown in Fig. 8. It shows how the voltage V_{out} converges to the voltage V_{target} as the weights are updated. The speed of the training process is described in Sec. 4.4.

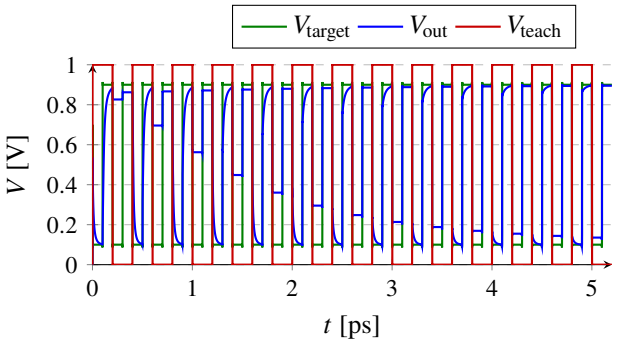

Fig. 8. Transient simulation of training process with dataset with two rows.

4.2 XOR Simulation

One part of verification is tested on AANN the XOR problem training, which is often used with analogue back-propagation [7], [15]. The simulations show how the training process of the designed neural network works. It is illustrated with the learning curves. As an example, an AANN with two inputs, eight neurons in a hidden layer and one output was simulated. The results of the analysis follow.

Each simulation was run ten times with different initial weights. At the end of each training epoch, the MSE was calculated. The results are shown in Figs. 9, 10 and 11, where each figure corresponds to a different choice of the learning rate η and each line of a different colour represents a different set of the initial network weights. It is possible to vary the learning rate by changing the value of V_{teach}, time given for training on one row of a dataset or other deep circuit parameters.

The simulations show that the proposed concept of AANN converges. The speed and stability depends on set of hyperparameters. Furthermore, classical neural networks behave in a similar way [7], [15].

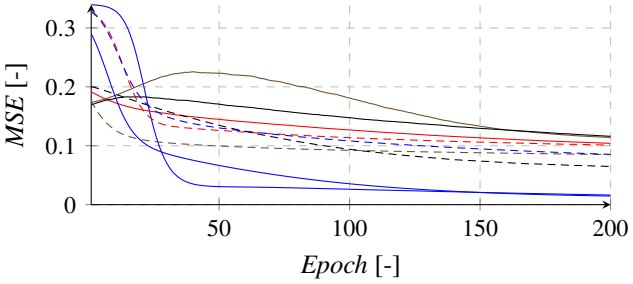

Fig. 9. AANN learning curve with small analogue learning rate (0.01 ps).

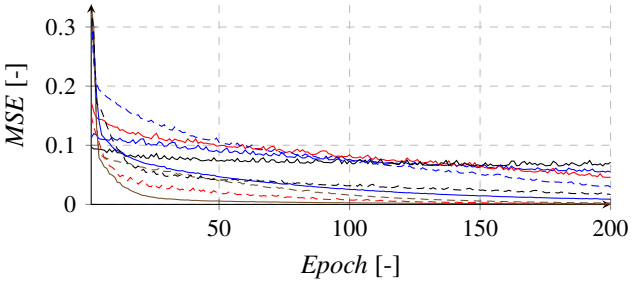

Fig. 10. AANN learning curve with optimal analogue learning rate (0.1 ps).

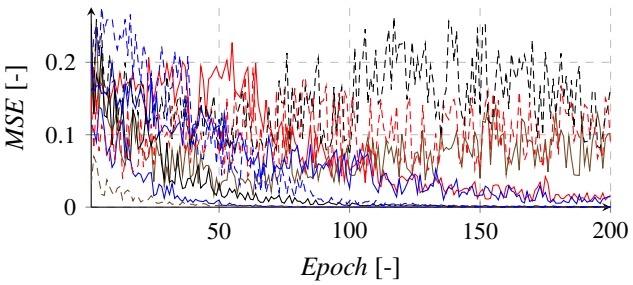

Fig. 11. AANN learning curve with big analogue learning rate (1 ps).

4.3 Dependence on Parasitic Properties

The whole article relies on the fact that the neural network can learn with its parasitic properties. However, this can work only to a certain extent. This simulation presents the toleration of this concept to some parasitic properties.

The block that most affects the scheme is the multiplier (Fig. 12). Inaccuracies are introduced into it according to

$$I_{\text{out}} = K_I \cdot \tanh(K_m \cdot V_{\text{in}_1} \cdot (V_{\text{in}_2} + V_{\text{off}})) \qquad (17)$$

where K_I is a constant determining the maximum current.

The result of this simulation is shown in Fig. 13. The comparison with Fig. 10 shows that the parasitic properties affect the proposed concept. However, it does not fundamentally affect its functionality.

Similar simulations such as noise resistance, the influence of non-linearity of CMOS capacitance, etc. can be demonstrated only partly with block design level, there are dependent on used implementation.

4.4 Speed Comparison

One of the main reasons to use a fully AANN instead of a classical one is the speed of training. This simulation shows a comparison between the proposed AANN and a neural network implemented with TensorFlow, an end-to-end open source platform for machine learning [12]. The computations are run on Central Processing Unit (CPU), Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU) hardware obtainable in Google Colab. [20] The NNs are trained and compared on the same datasets. The configurations are the most similar to the working flow of AANN described in this article. Unfortunately, the results of the other analogue implementations are not available for comparison [7], [17].

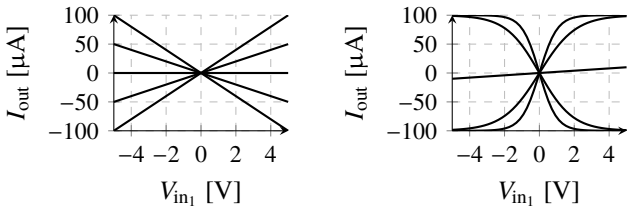

Fig. 12. The change of multiplier block for parasitic properties simulation.

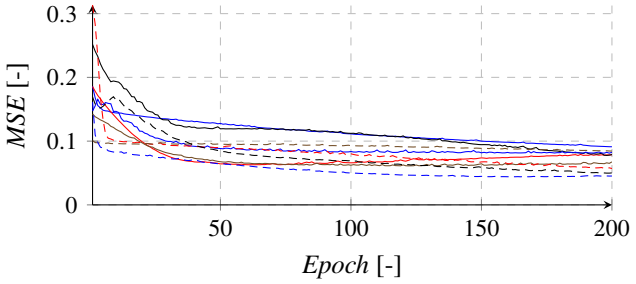

Fig. 13. AANN learning curve with changed multiplier block and learning rate (0.1 ps).

	Test 1	Test 2	Test 3	Test 4
CPU	2.017 ± 0.020 s	6.373 ± 0.097 s	54.678 ± 1.215 s	274.644 ± 2.281 s
GPU	4.872 ± 0.264 s	13.599 ± 0.314 s	120.152 ± 1.034 s	624.357 ± 6.211 s
TPU	2.017 ± 0.035 s	6.284 ± 0.190 s	53.910 ± 0.958 s	272.727 ± 2.159 s
AANN	2 ns	10 ns	100 ns	500 ns

Tab. 1. Comparison of time spent on training neural networks.

	Test 1	Test 2	Test 3	Test 4
Size of dataset	2	10	100	500
Number of hidden layers	1	2	3	4
Number of neurons	3	5	16	21
Number of epochs	1000	1000	1000	1000

Tab. 2. Size of the dataset and structure of neural networks for speed comparison.

Four different neural network structures were created, as is shown in Tab. 2. All these structures were created both in an analogue way and in a classical way. The result is shown in Tab. 1. The designed circuit still contains blocks defined on the formula level. This means that the result in real implementation is expected to be slower. Charging capacitors is the most time-consuming in this case. Values of capacitors and charging capacitors are depended on final hardware implementation. For example, in simulations, all capacitors have the value 0.1 pF and their maximal charged current is 100 μA per capacitor. These values will be changed according to specific implementation so that the parasitic properties do not affect the charging process. The block that is most prone to parasitic properties and therefore, the speed in this concept is the four-quadrant multiplier, which can tend to 40 GHz [21].

In AANN, the training time depends only on the size of the dataset and training time of one row of the dataset. For the classical neural network, the training time depends mainly on the structure of the network and not only on the size of the dataset.

The results show that potential real-time speed is around several orders of magnitude faster as compared to the known implementation.

5. Conclusion

This article introduces the concept of a novel fully analogue artificial neural network. It presents a block scheme for the fully-analogue backpropagation algorithm with stochastic gradient descent, which can be used for the supervised training of feedforward neural networks. This AANN was designed for the near-sensor application. It requires to be much faster and rather smaller than the classical neural networks and process analogue signals as inputs directly from sensors. This design is built on an idealized structure subjected to simulations. Simulations have shown that this block structure works and is around several orders of magnitude faster than the classical neural networks. It means that a real structure is expected to be significantly faster too.

Future work will be focused on the preparation of the transistor-level design. Furthermore, it will be possible to extend this design of AANN to the other structures, such as Autoencoder, Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Deep Neural Network (DNN), Long Short-Term Memory (LSTM) etc. From these structures, the CNNs seem to be the most useful option because they are used in applications at the limits of today's computer power.

Acknowledgments

This work has been supported by the grant No. SGS20/167/OHK3/3T/13 "Development of methods in electronics and signal processing" of the Czech Technical University in Prague.

References

[1] NARAYANAN, P., FUMAROLA, A., SANCHES, L. L., et al. Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory. *IBM Journal of Research and Development*, 2017, vol. 61, no. 4/5, p. 1–11. DOI: 10.1147/JRD.2017.2716579

[2] ROSENTHAL, E., GRESHNIKOV, S., SOUDRY, D., et al. A fully analog memristor-based neural network with online gradient training. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. Montreal (Canada), 2016, p. 1394–1397. DOI: 10.1109/ISCAS.2016.7527510

[3] PANTAZI, A., WOŚNIAK, S., TUMA, T., et al. All-memristive neuromorphic computing with level-tuned neurons. *Nanotechnology*, 2016, vol. 27, no. 35, p. 355205. DOI: 10.1109/ISCAS.2016.7527510

[4] SHAFIEE, A., NAG, A., MURALIMANOHAR, N., et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 2016, vol. 44, no. 3, p. 14–26. DOI: 10.1145/3007787.3001139

[5] SHI, W., CAO, J., ZHANG, Q., et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 2016, vol. 3, no. 5, p. 637–646. DOI: 10.1109/JIOT.2016.2579198

[6] KRESTINSKAYA, O., JAMES, A. P. Binary weighted memristive analog deep neural network for near-sensor edge processing. In *IEEE International Conference on Nanotechnology (IEEE-NANO)*. Cork (Ireland), 2018, p. 1–4. DOI: 10.1109/NANO.2018.8626224

[7] KRESTINSKAYA, O., SALAMA, K. N., JAMES, A. P. Learning in memristive neural network architectures using analog backpropagation circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2018, vol. 66, no. 2, p. 719–732. DOI: 10.1109/TCSI.2018.2866510

[8] VERLEYSEN, M., JESPERS, P. G. An analog VLSI implementation of Hopfield's neural network. *IEEE Micro*, 1989, vol. 9, no. 6, p. 46–55. DOI: 10.1109/40.42986

[9] MURRAY, A. F., DEL CORSO, D., TARASSENKO, L. Pulse-stream VLSI neural networks mixing analog and digital techniques. *IEEE Transactions on Neural Networks*, 1991, vol. 2, no. 2, p. 193–204. DOI: 10.1109/72.80329

[10] KRESTINSKAYA, O., SALAMA, K. N., JAMES, A. P. Analog backpropagation learning circuits for memristive crossbar neural networks. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. Florence (Italy), 2018, p. 1–5. DOI: 10.1109/ISCAS.2018.8351344

[11] CANCELO G., HANSEN S. Analog neural network development system with fast on line training capabilities. In *Proceedings of the Annual Conference of IEEE Industrial Electronics (IECON)*. Bologna (Italy), 1994, p. 1396–1400. DOI: 10.1109/IECON.1994.397999

[12] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning*. MIT press, 2016. ISBN: 978-0262035613

[13] RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint*, 2016. arXiv:1609.04747

[14] DITZLER, G., ROVERI, M., ALIPPI, C., et al. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 2015, vol. 10, no. 4, p. 12–25. DOI: 10.1109/MCI.2015.2471196

[15] MORIE, T., AMEMIYA, Y. An all-analog expandable neural network LSI with on-chip backpropagation learning. *IEEE Journal of Solid-State Circuits*, 1994, vol. 29, no. 9, p. 1086–1093. DOI: 10.1109/4.309904

[16] MIKHAILENKO, D., LIYANAGEDERA, C., JAMES, A. P., et al. M^2CA: Modular memristive crossbar arrays. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. Florence (Italy), 2018, p. 1–5. DOI: 10.1109/ISCAS.2018.8351112

[17] ERNST, O. K. *Stochastic Gradient Descent Learning and the Backpropagation Algorithm (Technical Report)*. University of California, San Diego, La Jolla, CA, 2014.

[18] PAULŮ, F., HOSPODKA, J. Web-based application for analysis of electrical circuits and systems. In *New Trends in Signal Processing (NTSP)*. Liptovsky Mikulas (Slovakia), 2018, p. 1–4). DOI: 10.23919/NTSP.2018.8524039

[19] CHIBLE, H., GHANDOUR, A. CMOS VLSI hyperbolic tangent function & its derivative circuits for neuron implementation. *International Journal of Electronics and Computer Science Engineering*, 2013, vol. 2, no. 4, p. 1162–1170. ISSN: 2277-1956

[20] WANG, Y. E., WEI, G. Y., BROOKS, D. Benchmarking TPU, GPU, and CPU platforms for deep learning. *arXiv preprint*, 2019. arXiv:1907.10701

[21] RAJPOOT, J., MAHESHWARI, S. High performance four-quadrant analog multiplier using DXCCII. *Circuits, Systems, and Signal Processing*, 2019, vol. 39, no. 1, p. 1–11. DOI: 10.1007/s00034-019-01179-x

About the Authors . . .

Filip PAULŮ was born in 1990. He received his Master's degrees from the Czech Technical University in Prague in 2016. Since 2016 he is a Ph.D. student at the Faculty of Electrical Engineering at the Department of Circuit Theory at the same university. His research is focusing on analogue artificial neural networks and his interest is to connect electrical engineering with artificial intelligence.

Jiří HOSPODKA was born in 1967. He received his Master's and Ph.D. degrees from the Czech Technical University in Prague in 1991 and 1995, respectively. Since 2007 he has been working as associate professor at the Department of Circuit Theory at the same university. Research interests: circuit theory, analog electronics, filter design, switched-capacitor, and switched current circuits.