Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 276 (2026) 223–237

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on System-Integrated Intelligence (SysInt 2025)
04 - 06 June 2025, Bremen, Germany

Design of Analog Computers: Building Blocks and Advanced Methods for Tiny Analog Machine Learning with Surrogate Modeling

Stefan Bosse[a]*

[a]University of Koblenz, Dept. of Computer Science, Chair of Practical Computer Science, 56070 Koblenz, Germany

Abstract

This study investigates the use of surrogate modeling of electronic circuits for functional optimization problems using gradient-based methods, demonstrated with an Analog Artificial Neural Network use-case. The results show that a surrogate model approach is more computationally expensive than a pure mathematical model (about 10 times), that the training process using error gradient back-propagation is instable and only randomly convergent, but it significantly improved classification error shown with the IRIS benchmark dataset, with the adam optimizer finding the "golden standard" solution with 2% classification error. The training results can be directly transferred to an analog transistor circuit without loss of accuracy.

© 2025 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 7th International Conference on System-Integrated Intelligence (SysInt 2025)

Keywords: Tiny ML; Analog Computers; Circuit Synthesis; Machine Learning; Analog Electronics; Simulated Annealing; Surrogate Model

1. Introduction

This study explores the shift from digital to analog computation using electronic circuits and investigates the use of surrogate machine learning models (ML) for designing analog circuits to perform numerical computations. The study focuses on the challenges of designing analog computers for the use case of Artificial Neural Networks (ANNs). Designing electronic circuits for specific mathematical models and ensuring stable and convergent parameterization of the model is complicated by Analog circuits which depend on environmental parameters like temperature and electronic component variations, which must be considered during the parameterization process.

* Corresponding author.
sbosse@uni-koblenz.de

1877-0509 © 2025 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 7th International Conference on System-Integrated Intelligence (SysInt 2025)

The study evaluates two methods for designing Analog ANN (AANN): simulation-in-the-loop ML training, which includes analog simulation of the target circuit in the training process and the loss function, and surrogate-models derived from simulation using digital ANN models, which provide much lower computational times. The integration of analog models in the training process increases the robustness of the final circuit against circuit and environmental variations and non-linearity of the analog components. A simple Structural Health Monitoring (SHM) example demonstrates the approach.

In the present study we address a paradigm shift from digital to analog computation using electronic circuits and we will investigate the deployment of surrogate ML models replacing analog simulation models for the design of such analog circuits performing numerical computations, finally demonstrated with an Analog Artificial Neural Network (AANN) enabling in-sensor computation.

Digital computing based on the binary number system is the standard for any numerical computation since about 60 years. Program-controlled computers are capable to perform highly complex numerical computations, e.g, solving of differential equations, with only a small set of instructions using high-level programming languages and compilers. With ongoing miniaturization, computation is integrated in sensors and devices towards material-integrated sensor networks (in-sensor computation). But the miniaturization towards the 1 mm^3 scale reduces computational power and memory capacity significantly.

Machine Learning is a prominent numerical problem class, where models already processed successfully on embedded systems, on some devices using limited integer arithmetic. But the resource constraints and the power consumption of digital embedded systems are still a limiting factor and a challenge, especially regarding material-integration of sensor nodes, e.g., used for SHM. Long forgotten analog electronic circuits can perform most numerical computations including neural network models [1] with a much lower amount of hardware resources and in parallel, with lower power consumption and they are not limited to integer arithmetic with its discretization and rounding errors. But analog circuits and computers, mostly composed of operational amplifiers, show non-linearity, drift, and other deviations from mathematical functions, moreover if organic or electrochemical transistors are used, and if the circuits is reduced to a minimal number of components as in this work. In this work we outline the challenges of the design of analog computers for the use case of an Artificial Neural Network (ANN) using different methods.

Why analog circuits? Some key facts:

- Resource requirements for the Sum-of-Product (SOP) computation: Digital system (Register-Transfer Logic) with more than 1000 transistors compared with analog systems with less than 10 transistors [2];
- Power consumption and energy efficiency for SOP computation (i.e., Multiply-Accumulate Computation MAC): Digital (Texas Instruments DSP): 4mw/s compared with analog (Field Programmable Analog Array, [3]): 4µW/s;
- Printed Organic Electronic with currently available 10-100 Transistors / mm^2: Digital: Not suitable, Analog: Suitable;
- Computational dynamic and resolution for any numerical function f(x): Digital discrete, 8-32 Bits, rounding errors up to 1% and underflow issues versa analog with continuous value distribution and dynamic range only limited by electronic noise;
- Start-up time: Digital: 100 ms (µC) versa analog nearly zero;
- Computational resources for solving differential equations (e.g., non-linear partial diff. eq., order 2, numerical computation requiring typically a high number of iterations): Digital systems requires roughly estimated 1s computational time, 1 MB RAM, more than 1 Millions of transistors (RTL) compared with analog systems with about 1ms delay and settling time, less than 100 transistors and 6 capacitors.

Two design goals must be addressed:

1. The design of electronic circuits for a specific mathematical model, e.g., an ANN, which should be implemented with analog electronics;
2. The stable and convergent parameterization of the model, in terms of ML this is the training process reducing the regression or classification error. Digital models need only mathematical stability, but analog circuits depend

additionally on environmental parameters like temperature and electronic component variations including aging, which must be considered during the parameterization process.

There is ongoing work to implement ANN with electronic circuits, mainly classifiable in using printed organic electronic circuits (application-specific) or silicon-based field programmable analog or transistor arrays (FPAA, FPTA) [2,3].

In previous work [4] we used a classical numerical model approach to train an AANN finally synthesized to an analog electronic circuit. Static and transient electronic simulation (using Spice3) was deployed to get parameters for the synthesis process and finally to post-design and to test these transformed analog circuits. The classical weighted sum-of-product and activation function model was modified to reflect analog circuit limitations, e.g., clipping and limited amplification, but electronic variations of the circuit were not considered during the digital training process. In this work, we will demonstrate the usefulness of surrogate ML models of an analog circuit that replace the analog simulation model for specific input-output relations which are required in the design and training process of analog ANNs (AANN).

There are basically three different approaches to train such an AANN:

1. Using a modified mathematical model of a neuron, performing error gradient descent parameter optimization, and finally converting the digital model to an analog circuit model, as discussed above (with low accuracy and a big reality gap);
2. Simulation-in-the loop ML training, i.e., including analog electronic simulation of the target circuit in the training process loop providing input for the loss function (but slowing down the training process significantly);
3. Surrogate models derived from electronic simulation using digital ANN models that are used in the training loop providing much lower computational times bit still reflecting the deviation of the used electronic circuit from the mathematical model (providing a good balance between accuracy and computational time).

We show the challenges and evaluate the third approach and compare the results with the first approach ([4]). If we cannot include the physical world or any exact model into our algorithms we need to create surrogate models to capture the most relevant features and behavior of the real world. A surrogate model is commonly a functional model, i.e., with well defined input and output variables and a directed data flow graph. An electronic circuit is an undirected cyclic graph of components. To create a surrogate model from an electronic circuit the data flow must be defined. Examples of surrogate modeling of electronic circuits using SVM models are shown in [5].

One important aspect of the integration of analog models in the training process of an ANN is the increase of robustness of the final circuit against circuit and environmental variations as well as non-linearity of the analog components, which should be investigated in this work. A simple SHM example use-case will demonstrate the usefulness of the proposed method.

2. The Analog Computer

If we perform computation on digital circuits, based either on application-specific Register-transfer Level Logic (RTL) or assuming an underlying programmable von Neumann computer architecture, we have to consider some aspects of the computational platform, basically the digital resolution, the memory capacity, and the memory architecture. If we want to perform computation on analog circuits we have to consider a significantly extended taxonomy, on the signal, design, and algorithmic level as shown in Figures 1 to 3. On signal level we have to distinguish static (quasi-stationary) and transient (dynamic) signals. The circuits can be divided into unipolar and bipolar circuits, with linear or non-linear transfer curves, small or large signal circuits and so on. In the time domain we can distinguish between low- and high-frequency circuits with or without a state (memory, history). On the design level we can create static design pure application-specific) or dynamically configurable architectures, either partial (parameters only) or fully (including the structure). Finally, on algorithmic level we can use a broad variety of algorithms to design the circuits from classical mathematical computations (like the starting point in this work) to stochastic approaches like Simulated Annealing or Generic Algorithms.

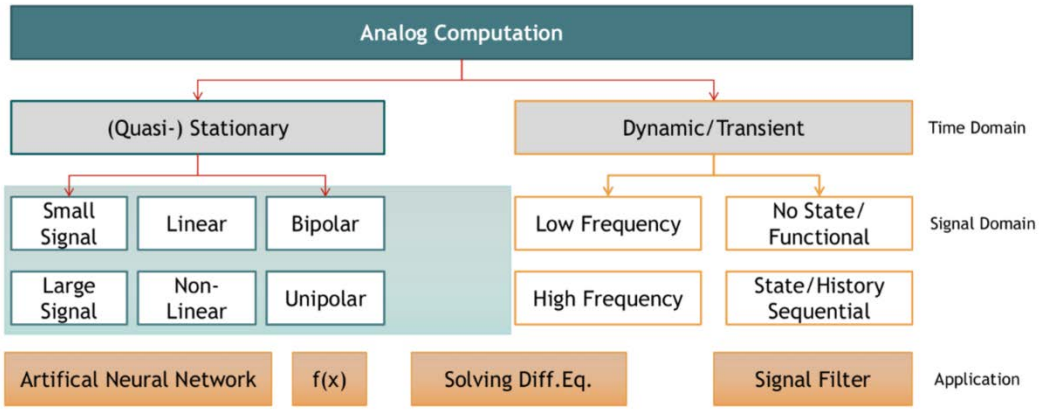

Fig. 1. Analog computer taxonomy: Time and signal domain with application examples

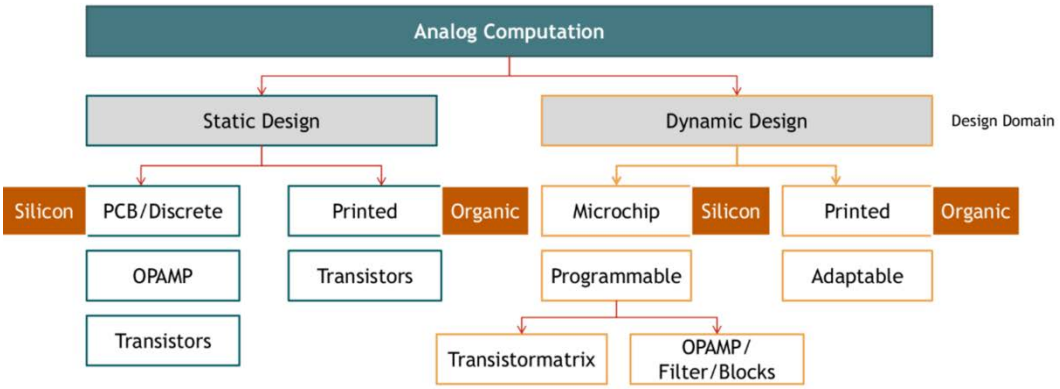

Fig. 2. Analog computer taxonomy: Technological design domain

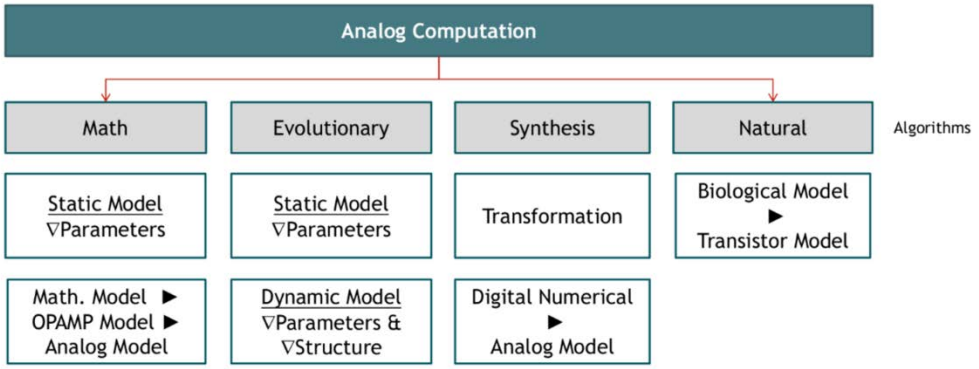

Fig. 3. Analog computer taxonomy: Algorithmic domain

The Operational Amplifier (OPAMP) is the basic cell of any accurate analog computational system. An OPAMP is basically a difference amplifier with an inverting and a non-inverting input i_+ and i_-, respectively. There is one output o. An OPAMP can implement a (perhaps time-dependent) function $y=f(x)$ by adding up to 7 functional blocks (e.g., resistive) defining the transfer function of the entire circuit, as shown in Fig. 4.

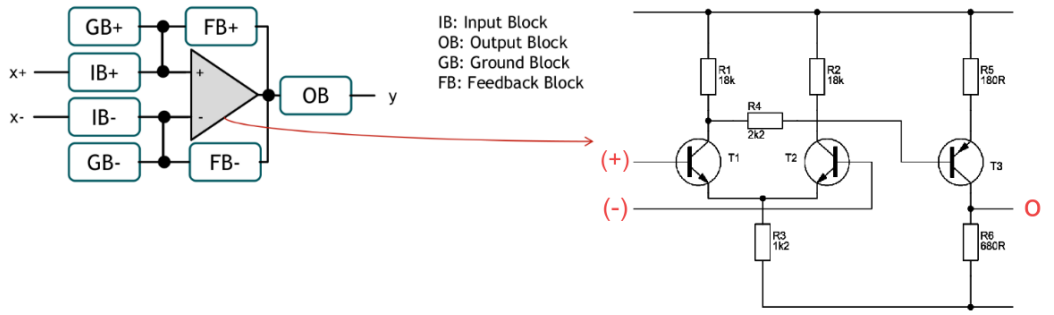

Fig. 4. (Left) OPAMP circuit block model (Right) Minimal transistor circuit of the difference amplifier (as used in this work)

3. AANN

The aim of this work is the implementation of a stationary (static) mathematical artificial neural network with analog electronics (AANN) and the investigation of suitable numerical training algorithms to adapt the parameters of the AANN to a specific problem. The AANN is an approximation of the mathematical ANN model (MANN).

Main limitations and deviations of an AANN compared to a MANN:

- Limited parameter ranges, i.e., weights and bias are limited within a certain range $[p_{min},p_{max}]$ due to limited open-loop gain of real electronic OPAMP circuits;
- Limited output range of the Sum-of-Products (SOP) block of a neuron within a certain range $[v_{min},v_{max}]$ due to limited supply voltages and saturation of electronic circuits;
- Non-linearity of the SOP and deviation of the activation function (here sigmoid) from the mathematical model (higher order non-linearity).

The AANN is composed of three OPAMP3 and one sigmoid block, as shown in Fig. 5 (see also [4]). Each OPAMP is implemented with a 3-transistor circuit creating an amplified difference amplifier. The open-loop gain of this OPAMP3 circuit is about 40. In a mathematical functional neuron model (see next section) there are parameters which can be positive or negative. The parameters are replaced here by resistors that set a specific amplification of the OPAMP3 circuit, i.e., a weight w=5 is equal to an input voltage amplification of k=5. Negative parameters cannot be implemented by such an electronic circuit, therefore a negative parameter is split in its sign and the positive multiplication factor. Depending on the sign of the parameter, a negative or positive amplification branch is required, a so called bipolar two-path architecture. Both paths are finally combined by another difference amplifier. For balance reasons it is not sufficient just to use one difference amplifier. The positive and negative input and functional blocks of an OPAMP show totally different "behavior".

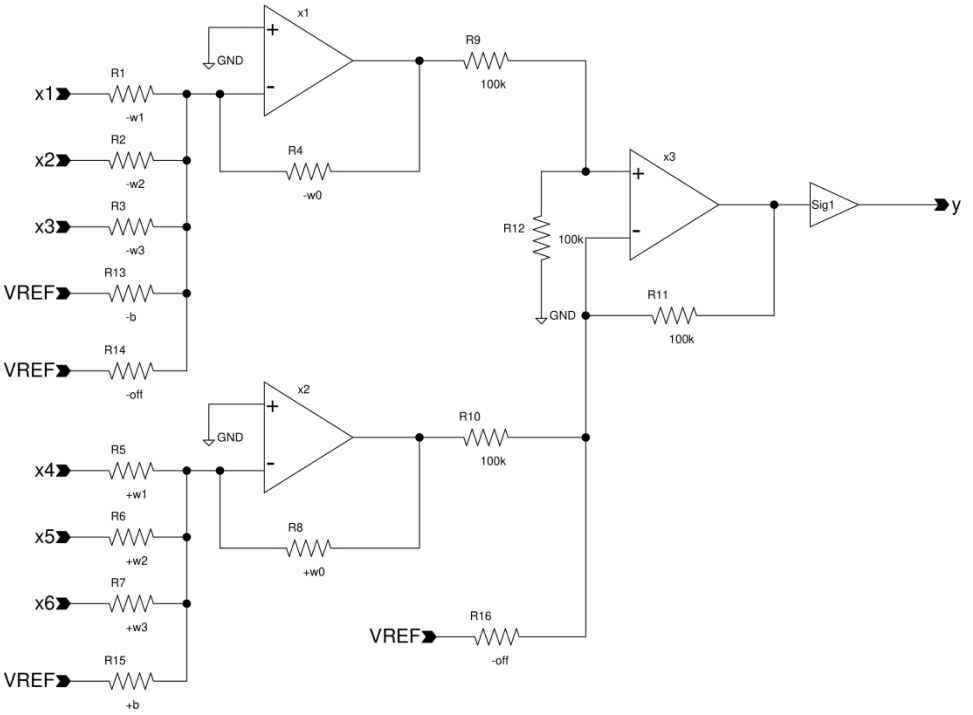

Fig. 5. Perceptron3 circuit: Bipolar analog neuron circuit. The inputs x_1-x_3 are negative weight inputs, whereas the inputs x_4-x_6 are positive weight inputs, VREF denotes a constant reference voltage, and y is the output port.

3.1. Mathematical Model

An ANN is composed of functional nodes (artificial neurons), given by vector function $f(\boldsymbol{x},\boldsymbol{w},\mathrm{b},\mathrm{a}):\boldsymbol{x} \rightarrow y$, mapping an input vector $\boldsymbol{x}$ on a scalar value y by using a set of parameters ($\boldsymbol{w}$,b) and a scalar transfer (activation) function a:

$$
\begin{aligned}
f\left(\vec{x},\vec{w},b\right) &= a\left(sop\left(\vec{x},\vec{w}\right)\right) \\
sop\left(\vec{x},\vec{w},b\right) &= \left(\sum_{i=1}^{|x|} w_i x_i\right) + b
\end{aligned}
\qquad (1.1)
$$

A fully connected ANN (FCANN) is composed of layers. Each layer i is populated with n_i functional nodes. Each node (except the first layer input nodes) are connected to all nodes of the previous layer. Neglecting numerical limitations there is an infinite input, parameter, and output space.

3.2. Clipped Mathematical Model

To reflect the limitations of real electronic circuits we introduced a bipolar modified mathematical model of a neuron (see [4] for details). There are independent paths for negative and positive weights, as shown in Fig. 6. The clipping ranges are set in accordance to electronic simulation results of the real perceptron3 circuit.

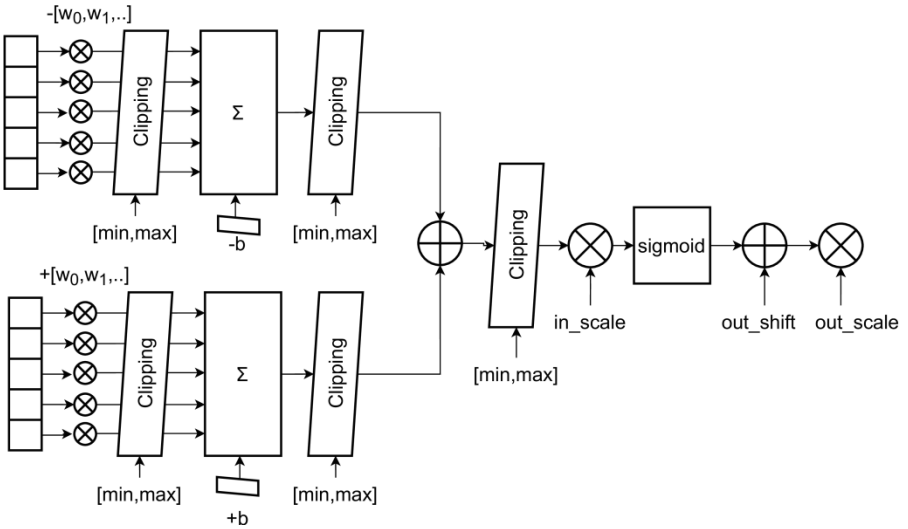

Fig. 6. The modified clipped mathematical/numerical ANN model (one neuron node, w_i are the weight parameters, b is a node offset)

In [4] an unipolar clipped neuron model was used. The parameter and SOP output clipping in conjunction with the unipolar model architecture introduced a reduced overall accuracy of the trained model (IRIS gold standard is typically 2% classification error, about 10% error was achieved).

3.3. Surrogate Model

The previously introduced modified mathematical perceptron model is still a rough approximation of the transfer function of a real electronic circuit, especially concerning the three-transistor approximation of the OPAMP model.

To capture the transfer characteristics with all its limitations of any real electronic circuit we created a surrogate model from simulation data of an electronic circuit with selected input-output assignments. An electronic circuit is just a cyclic undirected graph of functional nodes, e.g., with resistive, capacitive or non-linear characteristics. Only acyclic directed graphs can be represented by a pure functional surrogate model. We have chosen a differential Current-Controlled Voltage Source (CCVS) surrogate model of the perceptron circuit. Input are two currents i_+ and i_-, output is a scalar voltage value v_y. The original input resistor networks (representing the weight and bias parameters) are not part of the surrogate model. The complete model computation will assume linear resistor networks providing a positive and negative input current to the model, given by (for negative and positive parameters, respectively):

$$R_i = \frac{R_0}{|w_i|}, I_x = \sum_{i=1}^{n} \frac{x_i}{R_i} \tag{1.2}$$

The both current inputs are assumes as virtual grounds simplifying the current node computation. The basic CCVS surrogate model architecture and its embedding in the full perceptron circuit is shown in Fig. 7. The surrogate model is a simple three layer FCANN with tanh activation functions. The output scale is about 3, the input scale is about 1000.

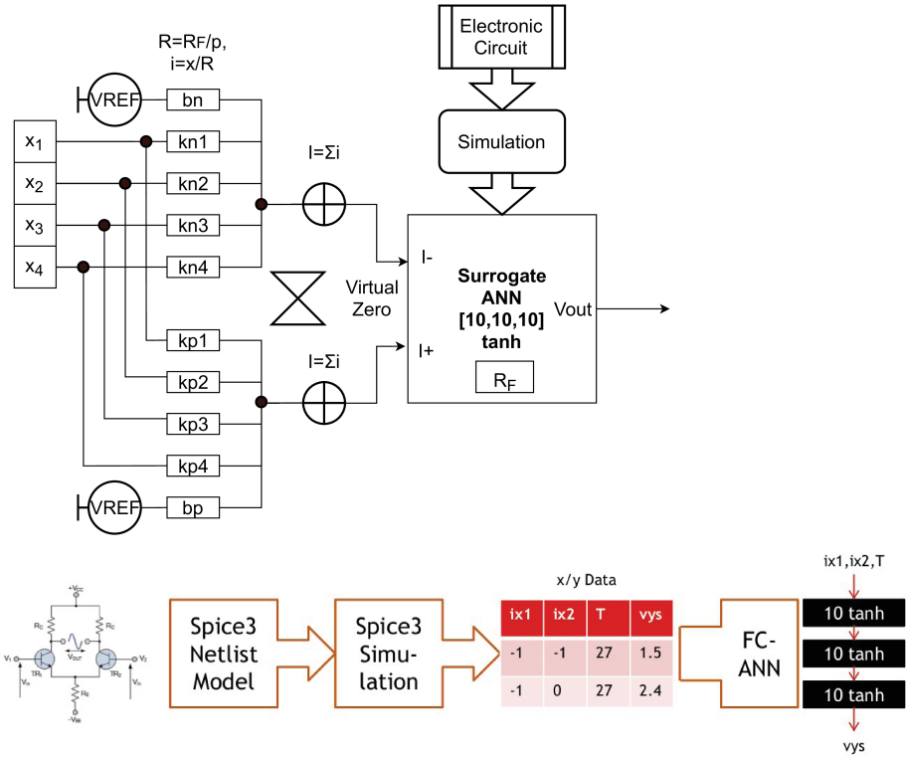

Fig. 7. (Top) A CCVS surrogate model of the perceptron core circuit. Important to note that the models is monolithic, i.e., there is no separation between the SOP and the activation function A. (Bottom) The workflow to create a surrogate model from a real electronic circuit via Spice electronic simulation and a 3-layer FCANN. The b/k resistors set the gain of the amplifier for the bias and weight parameters, respectively, R_F is a fixed feedback resistor, VREF a constant reference voltage.

In contrast to the original approach in [4] where a weight parameter either contributed to the positive or negative input path (exclusive or path), this model allows positive and negative contributions simultaneously (simple or path and a mixed union path).

The data-driven surrogate model is trained by data collected from electronic simulation (using the *ngspice* simulator). The simulation data spawns the parameter space, called training space. The comparison of the output of the surrogate with the simulation data is shown in Fig. 8. Within the interpolation range the S-model shows a very good accordance with the simulation data, but outside the parameter space in the extrapolation range strong deviation occur including oscillation of the transfer function. If we zoom in the left branch but still within the interpolation range we can see a small but relevant deviation. The electronic model is pure fallen monotonic, but the surrogate model shows varying monotonic behavior with changing sign of gradient. This non-monotonic behavior will have a significant impact on the convergence and stability of any gradient-based optimization algorithm, as shown in the results section.

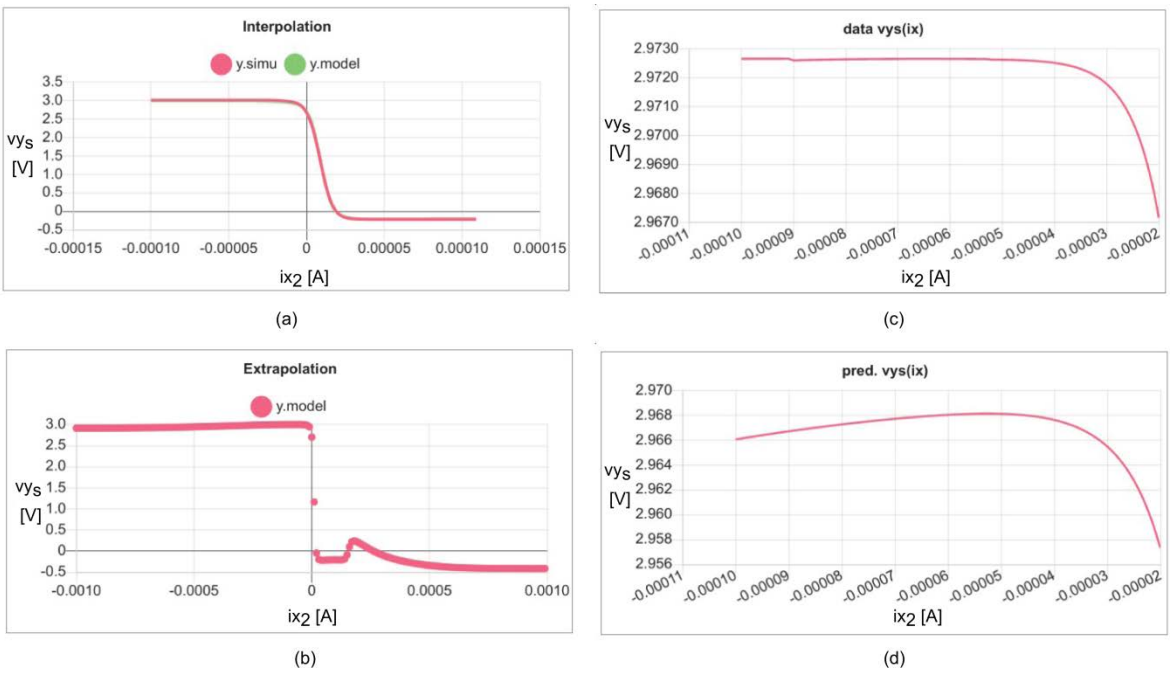

Fig. 8. Comparison of the output of the surrogate with the simulation data. (a) within training parameter range / interpolation (b) outside the training parameter range / extrapolation (c/d) zoom of the left branch / interpolation range, data and S-model, respectively. The first input variable was set to i_{x1}=0.

3.4. Meta Model

The surrogate model is part (basic cell) of a meta model that composes the ANN graph as shown in Fig. 9. The meta model is used for the training of AANN as well as for the final circuit synthesis, basically replacing the S-model with the electronic circuit and the parameters with the respective resistors of the SOP circuits. The parameters are given by the meta-model. The final circuit synthesis just requires the replacement of the S-model with the circuit components and the model parameters by resistors.

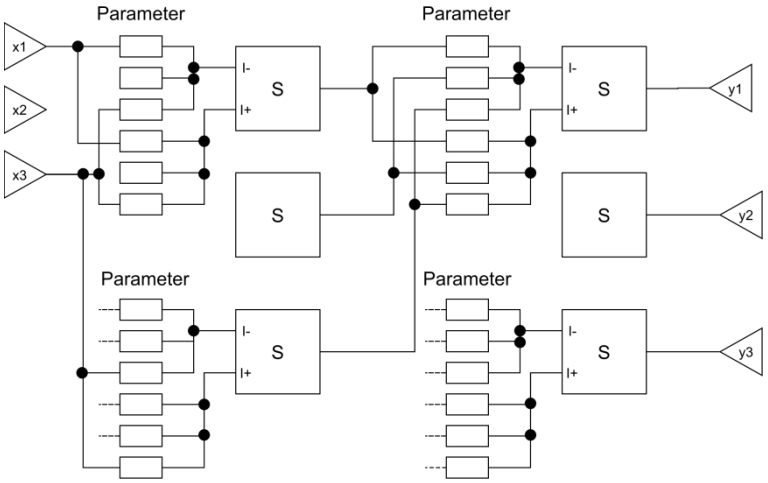

Fig. 9. Sketch of the structure of the ANN meta model using the surrogate model S for one sigmoid amplifier circuit combined with the external SOP circuit consisting of input resistors.

4. Error Gradient Descent Optimization

There are basically three model and training approaches to implement an ANN on an electronic circuit, as shown in Fig. 10:

1. Pure mathematical model trained with classical error gradient back-propagation, there are analytical differentiable functions $f \Rightarrow \partial f/\partial w$; no analog and electronic characteristics are considered;
2. A modified mathematical model considering analog electronic characteristics, e.g., clipping of function output and parameter, there are analytical differentiable functions $f \Rightarrow \partial f/\partial w$;
3. A surrogate model of the electronic circuit representing the analog behavior of the circuit; there are no analytical differentiable functions $f \Rightarrow \partial f/\partial w$; instead the surrogate model must be used to compute gradient numerically.

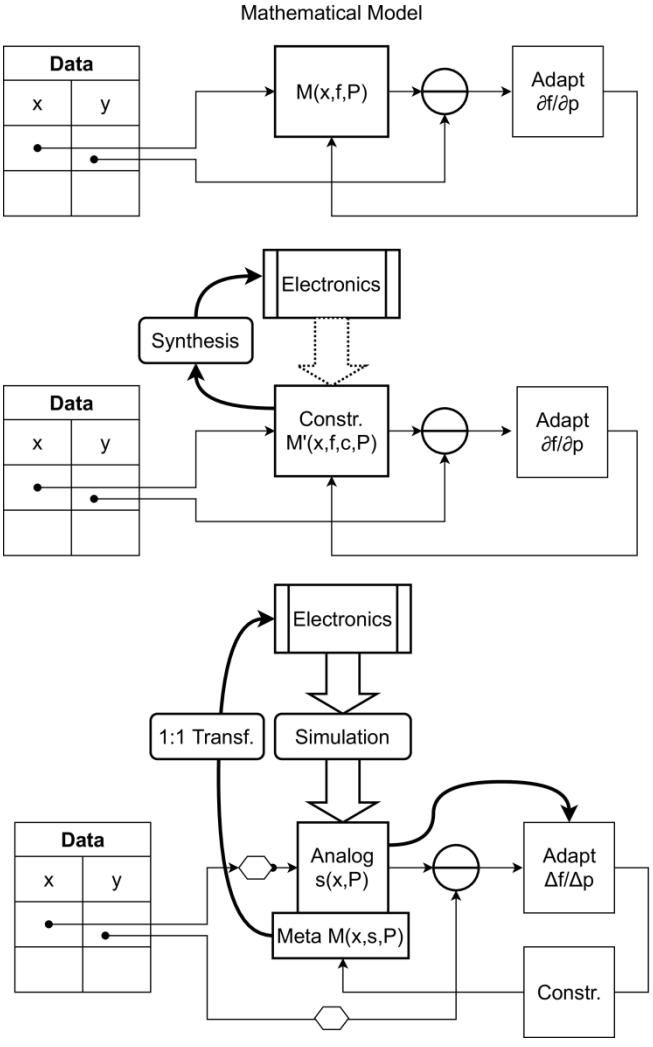

Fig. 10. Different methods to implement an ANN in an analog electronic circuit including data-driven training.

Any parameter updates of the bipolar model require a filtering with a *relu* function, i.e. allowing only positive parameters. Any negative parameter value is set to zero. This has an effect on the optimization convergence due to non-monotonic parameter updates.

In this work we compare different optimizer algorithms. The baseline is the classical error gradient back propagation. Assuming a parameterized function $f(\boldsymbol{x},\boldsymbol{w})$ the error gradient is computed by using a functional deviation with respect to one selected parameter w_i, which can be finally approximated by a numerical approximation (y_0 is the desired ground-truth output):

$$\frac{\partial err}{\partial w_i} \sim \frac{\Delta err}{\Delta w_i} = \frac{\Delta(y - y_0)}{\Delta w_i} = \frac{\Delta(f(x,\vec{w}) - y_0)}{\Delta w_i} = \frac{(f(x, w_i + \epsilon) - f(x, w_i) - y_0)}{\epsilon} \tag{1.3}$$

Computing the gradient with known mathematical functions *f* and analytical deviations is an easy task, but with complex functions like ANN (surrogate models) an analytical deviation is too complex. Therefore, a numerical gradient computation is used.

If we have the error gradient we can adjust each function parameter:

$$w' = w - \alpha \frac{\partial err}{\partial w} \tag{1.4}$$

The output error (which is just the difference between expected and actual output) must be back-propagated to the hidden layers just by summing up the weighted errors:

$$e_i = \sum_{j=1}^{J} e_j w_i, g_k = \frac{\partial f}{\partial w_k} e, g_k = e_i g_k \tag{1.5}$$

where *i* is the error of the i-th functional node in layer *n* and *j* the error from the functional node of the next layer *n*+1.

5. Simulated Annealing

The convergence of the error gradient back-propagation can be highly unstable, as shown in the results section. The algorithm can be caught in parameter space tales with a local error minimum. Even a momentum distortion cannot overcome this issue, especially in the case of non-monotonic transfer functions of a functional node. The success and convergence of a training run can depend only on the random initialization of the model parameters. To provide a parameter space pre-conditioning, the suitability of Simulated Annealing (SA) should be investigated, inspired by the work from [6]. The basic SA algorithm is shown in Alg. 1. The SA algorithm runs a fixed number of iterations. Instead of a utility function and a positive gain the model loss function and a negative gain is used to assess and improve the model parameters. The loss is computed for the entire training data set.

The advantage of SA-based model parameter pre-conditioning over error gradient back-propagation is the absence of the gradient computation required for the backward computation, resulting in a significantly reduced computational time. In this work the gradient is numerically computed and each parameter gradient computation requires two forward computations. To summarize, the backward computation requires 2*k* forward computations for *k* parameters per node. The factor 2 originates in the bipolar model architecture. This increase of the computational time is not significant for a simple mathematical model, but significant if a computational intensive surrogate model is used (here a three-layer FCANN with 10 nodes per layer).

```
function SA(data, params, num_steps=1000, noise=0.01, cooling=0.999) {
  initial_params=optimal_params=best_params=params; temp=1.0
  new_loss = best_loss = loss(data,params)
  for(i=1,num_steps) {
    temp=temp*cooling
    new_params = params.map(p => max(0,p+gaussianRandom(0, noise)))
    new_loss = loss(data,new_params)
    if (new_loss < best_loss ||
        random()*temp > best_loss/new_loss) {
       params = new_params
       if (new_loss < best_loss) {
         optimal_params = params;
         best_loss = new_loss
       }
    }
  }
}
```

Alg. 1. Simulated Annealing algorithm. Any parameter update of the bipolar model requires a filtering with a *relu* function, i.e. allowing only positive parameters. Any negative parameter value is set to zero.

6. Results and Discussion

Different training methods were evaluated including Simulated Annealing (SA). The core parameters of the training algorithms are shown in Tab. 1. We applied the training methods to the bipolar clipped mathematical model as well as to a meta-model using the surrogate model derived from electronic circuit simulation.

Method	Description	Parameters
SA	Simulated Annealing	Epochs=2000, cooling rate=0.999, noise amplitude=0.001
gd	Descent gradient error back-propagation	momentum=0, learning rate=0.01/0.001
sgd	Stochastic gradient descent error back-propagation	momentum=0.9, learning rate=0.01/0.001
adadelta	Adaptive learning rate	learning rate=0.01/0.001, eps=1e-8
adagrad	Adaptive Gradient	eps=1e-8, ro=0.95
nesterov	Accelerated gradient	momentum=0.9, learning rate=0.01/0.001
adam	Adaptive first- and second order moments	learning rate=0.01/0.001, beta1=0.9, beta2=0.999, eps=1e-8

Tab. 1. Training methods and their parameters compared in this work. If not other specified: $Epochs_{max}$=200, class.error threshold=0.05,

Note that the classification error threshold is compared with the validation classification error after one epoch of training (i.e., after dynamic model variations). The dynamic classification error calculated during model updates can be different.

One important observation is the bipolar parameter split quality. With the unipolar clipped mathematical model [4] a parameter was assigned either to the negative or positive perceptron input (exclusive or), the bipolar clipped mathematical model shows mostly this sign split (nearly exclusive or). With the bipolar electronic surrogate model this is not always the case. Sometimes positive and negative parameters have significant contributions.

Tables 2 and 3 compare the training results for the clipped mathematical and the electronic surrogate model with respect to different training methods (and with and without Simulated Annealing applied to pre-condition the model parameter set). The SA parameter pre-conditioning has no significant or sightly negative impact on the training process using the bipolar clipped mathematical model. Neither the classification error nor the epoch statistics are changed. The SA method has a slight impact if using the electronic surrogate model typically reducing the average number of epochs of about 5-10% and a reduction of the classification error. The *adagrad* optimizer always fails (in both model classes). The adam optimizer outperforms all other optimizer with the mathematical model, but not with the surrogate model. Here *adadelta* showed the lowest minimal and average number of epochs. The *adam* optimizer creates the highest model overflow rate *OV* (i.e., using the model outside the interpolation parameter range), the lowest OV rate is achieved all other optimizers. With SA the chance is higher to find the best fit (classification error of 1.99%). The simple *gd* optimizer (without momentum) is competitive and delivers comparable results.

The main difference between the bipolar clipped mathematical model and the surrogate model can be seen in the training convergence and stability. Although, clipping (relu function) has an impact of gradient optimization approaches, the standard deviation of the number of epochs and the final classification error is relative low. In contrast to the usage of the surrogate model (with its non-monotonic transfer behavior). The standard deviation is significantly increased and a successful training termination (meeting the error threshold) is only a probabilistic process and non-deterministic.

M	SA	ep_{min}	ep_{mean}±sd	E_{min} [%]	E_{mean}±sd [%]	FT [µs]	BT [µs]
gd	0	79	111.48 ± 16.03	3.97	4.59 ± 0.17	6.77	303.41
gd	1	82	106.75 ± 16.24	3.97	4.59 ± 0.17	16.57	302.92
sgd	0	78	110.84 ± 14.81	3.97	4.6 ± 0.14	16.85	306.74
sgd	1	78	110.48 ± 15.56	3.97	4.58 ± 0.19	17.06	308.94
adagrad	0	200	200 ± 0	52.32	97.85 ± 8.59	16.7	300.41
adagrad	1	200	200 ± 0	100	100 ± 0	16.75	299.58
adadelta	0	200	200 ± 0	33.11	33.64 ± 0.39	16.86	302.73
adadelta	1	200	200 ± 0	33.11	33.7 ± 0.43	17.06	307.82
nesterov	0	81	109.68 ± 15.12	3.97	4.62 ± 0.11	17.16	306.07
nesterov	1	77	108.17 ± 12.51	3.97	4.61 ± 0.13	16.93	304.88
adam	0	9	21.68 ± 4.81	2.65	3.92 ± 0.57	17.38	305.8
adam	1	14	22.85 ± 5.77	2.65	4.01 ± 0.57	18.12	310.47

Tab. 2. Training results for the clipped bipolar mathematical AANN model using different training algorithms and with and without SA parameter pre-conditioning. ep: Number of epochs, E: Classification error, FT: Forward time, BT: Backward time, sd: standard deviation (n=100 test runs).

M	SA	ep_{min}	ep_{mean}±sd	E_{min} [%]	E_{mean}±sd [%]	OV [%]	FT [µs]	BT [µs]
gd	0	11	146.7 ± 68.25	2.65	29.5 ± 29.18	0.02	106.9	2181.93
gd	1	11	141.66 ± 73.8	2.65	30.93 ± 31.16	0.02	104.66	1794.74
sgd	0	10	144.53 ± 72.58	3.31	30.51 ± 30.01	0.02	110.74	2185.22
sgd	1	6	143.42 ± 76.81	1.99	28.44 ± 26.08	0.01	109.89	2023.24
adagrad	0	32	198.32 ± 16.72	3.97	61.42 ± 25.33	0.02	91.83	2046.93
adagrad	1	7	198.07 ± 19.2	4.64	51.54 ± 22.89	0.02	104.45	1848.4
adadelta	0	4	110.44 ± 82.86	2.65	16.92 ± 15.88	0.08	106.09	2028.94
adadelta	1	7	85.84 ± 73.76	2.65	13.92 ± 18.91	0.06	111.88	1948.48
nesterov	0	9	158.45 ± 66.43	3.31	34.25 ± 30.17	0.02	112.37	2221.83
nesterov	1	8	129.08 ± 81.36	3.31	30.79 ± 31.89	0.01	111.19	2054.2
adam	0	2	107.2 ± 82.78	2.65	18.63 ± 24.96	0.17	112.96	2415.1
adam	1	0	99.17 ± 88.98	1.99	16.34 ± 18.8	0.2	114.52	2175.49

Tab. 3. Training results for the clipped mathematical AANN model using different training algorithms and with and without SA parameter pre-conditioning. ep: Number of epochs, E: Classification error, OV: model overflow rate (input out of range), FT: Forward time, BT: Backward time, sd: standard deviation (n=100 test runs).

The increased computation time of the surrogate model results in an 8 times higher forward and backward computation times. The backward computation time computing the function gradients requires 16 times higher computation time. The advantage of the SA parameter pre-conditioning is the lower computational time since no backward computation must be carried out. But SA still not results in a better convergence rate.

Finally, the best trained meta-model (using the electronic bipolar surrogate model) was transformed into a transistor circuit. The simulation of this circuit showed the same behavior and finally classification error (2%).

7. Conclusions

In this work the surrogate modeling of electronic circuits for solving functional optimization problems using gradient-based methods were investigated. The functional optimization problem was demonstrated with an analog artificial neural network circuit implementing a classification problem. The results were compared with the deployment of a pure mathematical model with respect to computational times and final classification error. The approximated numerical function deviation is much more computational expensive compared with the mathematical deviation and computation of a simple function (about 8 times slower). The bipolar model requires the double number of forward and backward computations, which additionally increases the computation time. To enable training of generic AANN a bipolar architecture were chosen. In previous work [4] a clipped unipolar mathematical model was trained with classical gradient-based algorithms. The signed parameters were finally mapped on a bipolar OPAMP-based perceptron circuit. In this work the bipolar architecture was trained directly, either by using a bipolar clipped mathematical model or by using the electronic surrogate model of the perceptron circuit. In contrast to the mathematical model the usage of the surrogate model introduced a highly instable and only random convergent training process, regardless of the used optimization algorithm. Simulated Annealing was tried to improve the model parameter pre-condition, but with low benefit. A monitored back-tracking algorithm was used to find a suitable solution. In contrast to [4] there was a significant improvement of the classification error shown with the IRIS dataset by using the surrogate model approach. The adam optimizer required the lowest average epoch runs and found the "golden standard" solution with 2% classification error.

References

[1] Ji, J., Gao, D., Wu, H. Y., Xiong, M., Stajkovic, N., Latte Bovio, C., and S. Fabiano (2025). “Single-transistor organic electrochemical neurons.” *Nature Communications*, **16**(1): 4334-4338.

[2] Weller, D. D., Bleier, N., Hefenbrock, M., Aghassi-Hagmann, J., Beigl, M., Kumar, R., and M. B. Tahoori. (2021, February). “Printed stochastic computing neural networks.” In 2021 *Design, Automation & Test in Europe Conference & Exhibition* (DATE) 914-919.IEEE

[3] Hasler , J., (2024) “Energy-Efficient Programable Analog Computing: Analog computing in a standard CMOS process.” IEEE *Solid-State Circuits Magazine*, **16**(4)

[4] Bosse , S and B. Lüssem, (2024) “Analog Electronics Neural Networks: Analog Computing combined with Digital Data Processing Revisited.” in *Proceedings of the 11th International Electronic Conference on Sensors and Applications*, 26–28 November 2024, MDPI: Basel, Switzerland, doi:10.3390/ecsa-11-20463

[5] Ciccazzo, A., Pillo, G. D., and V. Latorre (2014).“Support vector machines for surrogate modeling of electronic circuits-”. *Neural Computing and Applications*, **24**: 69-76

[6] Koza, J. R., Bennett, F. H., Lohn, J., Dunlap, F., Keane, M. A. and D. Andre. (1997, April). “Automated synthesis of computational circuits using genetic programming.” In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation* (ICEC'97) 447-452 IEEE.