sensors

MDPI

Article

CMOS Implementation of ANNs Based on Analog Optimization of N-Dimensional Objective Functions

Alejandro Medina-Santiago [1,*], Carlos Arturo Hernández-Gracidas [2,*], Luis Alberto Morales-Rosales [3], Ignacio Algredo-Badillo [1], Monica Amador García [4] and Jorge Antonio Orozco Torres [5]

1 Department of Computer Science, CONACYT-INAOE (Instituto Nacional de Astrofísica, Óptica y Electrónica), Santa María Tonanzintla, Puebla 72840, Mexico; algredobadillo@inaoep.mx
2 Physical-Mathematical Science Department, CONACYT-BUAP, Puebla 72570, Mexico
3 Faculty of Civil Engineering, CONACYT-Universidad Michoacana de San Nicolás de Hidalgo, Morelia 58000, Mexico; lamorales@conacyt.mx
4 Instituto Tecnológico Superior de Rioverde, Tecnológico Nacional de México, San Luis Potosi 79610, Mexico; mony_951@hotmail.com
5 Campus Tuxtla Gutiérrez, Tecnológico Nacional de México, Tuxtla Gutiérrez 29050, Mexico; jorge.ot@tuxtla.tecnm.mx
* Correspondence: amedina@inaoep.mx (A.M.-S.); cahernandezgr@conacyt.mx (C.A.H.-G.)

Abstract: The design of neural network architectures is carried out using methods that optimize a particular objective function, in which a point that minimizes the function is sought. In reported works, they only focused on software simulations or commercial complementary metal-oxide-semiconductor (CMOS), neither of which guarantees the quality of the solution. In this work, we designed a hardware architecture using individual neurons as building blocks based on the optimization of n-dimensional objective functions, such as obtaining the bias and synaptic weight parameters of an artificial neural network (ANN) model using the gradient descent method. The ANN-based architecture has a 5-3-1 configuration and is implemented on a 1.2 µm technology integrated circuit, with a total power consumption of 46.08 mW, using nine neurons and 36 CMOS operational amplifiers (op-amps). We show the results obtained from the application of integrated circuits for ANNs simulated in PSpice applied to the classification of digital data, demonstrating that the optimization method successfully obtains the synaptic weights and bias values generated by the learning algorithm (Steepest-Descent), for the design of the neural architecture.

Keywords: CMOS circuit; analog system; signal processing; learning algorithm; artificial neural network

check for updates

Citation: Medina-Santiago, A.; Hernández-Gracidas, C.A.; Morales-Rosales, L.A.; Algredo-Badillo, I.; Amador García, M.; Orozco Torres, J.A. CMOS Implementation of ANNs Based on Analog Optimization of N-Dimensional Objective Functions. *Sensors* **2021**, *21*, 7071. https://doi.org/10.3390/s21217071

Academic Editors: Jong-Ryul Yang and Seong-Tae Han

Received: 30 September 2021
Accepted: 19 October 2021
Published: 25 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The design process of complementary metal-oxide-semiconductor (CMOS) circuits consists of defining circuit inputs and outputs, hand calculations, circuit simulations, circuit layout, simulations including parasitics, reevaluation of circuit inputs and outputs, fabrication, and testing [1]. Circuit specifications are rarely defined; they can change as the design of the circuit or application progresses. This is the result of seeking the reduction of costs and improving performance of the design in its manufacture; it can also be due to the chip type or needs of the end-user. In most cases, it is not possible to make major changes to the design once the chip is in production (www.mosis.org, accessed on 18 October 2021). The characteristics of the CMOS allow the integration of logic functions with high density in integrated circuits. Due to this, CMOS has become the most widely used technology within very large-scale integration (VLSI) chips [2,3].

CMOS is used in static random access memory (RAM), digital logic circuits, microprocessors, microcontrollers, image sensors, and the conversion of computer data from one file format to another. Most configuration information on newer central processing units (CPUs) is stored on one CMOS chip. The configuration information on a CMOS chip is

called the real-time clock/nonvolatile RAM (RTC/NVRAM) chip, which works to retain data when the computer is shut off.

CMOS transistors are very famous because they use electrical power efficiently. They use no electrical supply whenever they are alternating from one condition to another. Furthermore, the complementary semiconductors work mutually to stop the o/p voltage. The result is a low-power dissipation VLSI design; for this reason, these transistors have changed other earlier designs like charge-coupled devices (CCDs) within camera sensors, which are used in most of the current processors. CMOS memories within a computer are a kind of non-volatile RAM that store BIOS settings, as well as time and date information.

Although most of the advances in neural networks have resulted from theoretical analysis or computer simulations, many of the potential advantages of artificial neural networks (ANNs) are expected to be implemented in hardware. Fortunately, the rapid advance in VLSI technology has made many of the previously impossible ideas now feasible to realize. Therefore, in this work, the numerical analysis of an n-dimensional objective-function optimization strategy is presented. This strategy, based on gradient descent, and its software implementation for the optimization of the weight and bias parameter values in multilayer ANNs, with the purpose of achieving their convergence, are presented as well [1,4,5]. A hardware implementation using CMOS transistors with operational amplifiers as its base elements is designed from the resulting ANN training model. A case study for the classification of digital patterns using multilayer ANNs is presented, which illustrates the implementation of the whole process. The electronic simulation of the developed integrated circuit in PSpice shows it is feasible to consider that this optimization method is efficient. In summary, the contributions of this work are as follows, sorted by relevance:

1. The design of circuits, using CMOS transistors through operational amplifiers as base elements, of the training model obtained from the parameter optimization, applied to multilayer ANNs.
2. An analysis of the designed integrated circuit (ANN hardware) based on the electronic simulation results in Pspice.
3. The software simulation of the weight and bias parameter optimization, with the purpose of achieving the convergence of a multilayer ANN (the automatic learning algorithm) based on gradient descent.
4. A multilayer ANN hardware case study simulation, designed to classify digital patterns to identify whether data is correctly sent.

Optimization techniques are ubiquitous in the field of ANNs. Generally, most learning algorithms used for training ANNs can be formulated as optimization problems because they are based on the minimization of an error function (Lyapunov function), called $E(x)$, where $x = [x_1, x_2, \ldots, x_n]^T$ is the parameter vector. One of the most important fundamentals required in the learning algorithms is the calculation of the gradient of a specific objective function $\nabla E(x)$ with respect to these parameters (synaptic weights w_{ij}, gain parameter γ_i, decay constant characteristic α_i, and Lagrange multipliers) [6–8].

It is well known that almost all optimization problems are solved numerically by iterative methods. Some of these iterative methods can be considered as a discrete-time realization of continuous-time dynamic systems. Specifically, a continuous-time (analog) dynamic system is described by a set of ordinary differential Equations (usually nonlinear), while a discrete-time dynamic system involves systems of differential equations. The advantages of using a system of differential equations are:

1. Real-world problems can be modeled with the use of differential equations, which allow humans to understand various fundamental laws of science [7]. Therefore, the simulation or implementation of a system of differential equations allows real-time optimization problems to be solved; this is due to the extraordinary parallel operation of the calculation units and the convergence properties of the neural systems.
2. The convergence properties of a continuous-time system are better because the learning rates can be set to be arbitrarily large without affecting the system stability. In con-

trast, in a discrete-time system, we must bound the control parameters in a small interval or, otherwise, the system may become unstable (i.e., the algorithm diverges).
3. A dynamic system implemented with basic differential equations exhibits more robustness to certain parameter variation [9,10].
4. Sometimes, continuous-time systems link different discrete-time iterative systems, as special cases for discretization, leading to the development of new iterative algorithms. Consequently, the simulation of continuous-time dynamic systems is more sophisticated and has faster techniques.

This work is organized as follows. Section 2 of this article shows the mathematical foundations of the Steepest-Descent objective function optimization method. Section 3 shows the analysis and development of the proposed model for the implemented circuit. Section 4 exemplify the design of integrated circuits for ANN architectures with CMOS transistors. Section 5 shows the application of the ANN to digital pattern recognition. Section 6 presents a comparison between ours and other related works. Finally, Section 7 expresses the conclusions of the work.

2. Mathematical Foundations

The basic principles of optimization were discovered in the 17th century by scientists and mathematicians such as Kepler, Fermat, Newton, and Leibniz [11,12]. Since 1950, these principles have been rediscovered for their implementation in digital computers. The progress of this effort significantly stimulated the search for new algorithms, and the field of optimization theory came to be recognized as one of the best fields of mathematics. Currently, researchers using ANNs have access to a wide range of theories that can be applied to the training of these networks.

A well-known mathematical algorithm used for solving the optimization problems is described below: the basic iterative Steepest-Descent algorithm [13,14]. The top-down method, also known as the gradient method, is defined in an n-dimensional space and is one of the oldest techniques used for minimizing a given cost function. This method forms the basis of the direct methods used in the optimization of restricted and unrestricted problems. Furthermore, it is the most used technique for non-linear optimization. In other words, what is sought is a value of x that minimizes the cost function $E(x)$.

Let us consider the following unconstrained optimization problem: find a vector $x \in \mathbb{R}^n$ that minimizes the real-valued scalar function

$$E = E(x), \tag{1}$$

This function is called the cost, objective, or energy function and x is an n-dimensional vector called the design vector. Minimizing a function is the same as maximizing the negative of the function, so there is no loss of generality in our considerations.

The point x^* is a global minimizer for $E(x)$ if $E(x^*) \leq E(x)$ for all $x \in \mathbb{R}^n$, and a strict local minimizer if the relation $E(x^*) \leq E(x)$ holds for a ball $B(x^*; \epsilon)$.

Assuming that the first and second derivates of $E(x)$ exist, a point x^* is a strict local minimizer of $E(x)$ if the gradient is zero (i.e., $\nabla E(x^*) = 0$) and the Hessian matrix is positive definite (i.e., $x^T \nabla^2 E(x^*) x > 0$).

The above statement can be formulated as a theorem on necessary and sufficient conditions for a strict local minimizer: Let $\nabla^2 E(x)$ be nonsingular for point x^*. Then, we have $E(x^*) < E(x)$ for every x in $0 < ||x - x^*|| < \epsilon$ with some $\epsilon > 0$, if $\nabla E(x^*) = 0$ and $\nabla^2 E(x^*)$ is symmetric and positive definite.

Considering an optimization problem, which has a cost function:

$$E(x) \text{ subject to } x \in \mathbb{R}^n, \tag{2}$$

Suppose there exists, at the same time, a gradient vector and the Hessian matrix of the objective function (i.e., they can be evaluated analytically), then this involves generating a sequence of search points $x^{(k)}$ through the iterative procedure:

$$x^{(k+1)} = x^{(k)} + \eta^{(k)} * d_k,\ x(0) = x^{(0)},\ (k = 0, 1, 2, \ldots), \tag{3}$$

where $\eta^{(k)} > 0$ determines the length of the step (learning rate) to be taken in the direction of the vector $d_k \cong \Delta x^{(k)}$ (search direction). In numerical optimization, there are different techniques to calculate the parameter $\eta^{(k)}$ and the direction of the vector d_k. For convenience, four basic methods are presented:

1. Gradient method (Steepest-Descent) [6,12,15], where the direction is defined as:

$$d_k := -\nabla E(x^{(k)}), \tag{4}$$

2. Newton's method [11], in which the search direction is determined by:

$$d_k := -[\nabla^2 E(x^{(k)})]^{-1} \nabla E(x^{(k)}), \tag{5}$$

3. Since the calculation of the Hessian inverse matrix $[\nabla^2 * E(x^{(k)})]^{-1}$ can be slightly complicated, a symmetric positive matrix of dimension $n \times n$ is defined, which is called H_k.

$$H_k \cong [\nabla^2 E(x^{(k)})]^{-1}, \tag{6}$$

Applying the quasi-Newton method [11,12,15] the search direction is determined by:

$$d_k = -H_k \nabla E(x^{(k)}), \tag{7}$$

4. The conjugate gradient method [11,12,15–17] calculates the current search direction d_k as a linear combination of the current gradient vector and the previous search direction. This simple way to find the direction is calculated by:

$$d_k = -\nabla E(x^{(k)}) + \beta_k d_{k+1},\ (k = 0, 1, 2, \ldots), \tag{8}$$

with $d_0 = -\nabla E(x^{(0)})$, where β_k is a scalar parameter that ensures that the vector sequence d_k satisfies the condition of mutual conjugation.

The length of the step $\eta^(k)$ is usually determined by these methods using one of the following techniques: (1) Minimization along the line, and (2) A fixed step size in one dimension. In the next subsection, the mathematical development of the continuous-time interactive algorithm for calculating the local minimum of a cost function is described.

2.1. Continuous-Time Iterative Algorithm

The gradient descent method discussed above can be written as:

$$x^{(k+1)} := x^{(k)} - M_k \nabla E(x^k), \tag{9}$$

where M_k is symmetrically defined as a positive matrix of dimension $n \times n$. The appropriate choice of matrix M_k is critical, based on the convergence properties of the algorithm. The discrete-time minimization algorithm determines the local minimum of the cost function $E(x)$ as the limit of the sequence $E(x^{(k)})(k = 0, 1, 2, \ldots)$ where $x^{(0)}$ is an initial estimate of the local minimizer X^*.

These iterative algorithms generate a sequence of points $\{x^{(k)}\}$ and a search direction $d_k \cong \Delta x^{(k)}$ through a discrete-time approximation for some continuous-time trajectory

from the starting point $x^{(0)}$ to the minimum point x^* (stationary). The continuous-time trajectory $x(t)$ is usually determined by a system of differential equations as follows:

$$\frac{dx_j}{dt} = -\sum_{i=1}^{n} \mu_{ji} \frac{\partial E(x)}{\partial x_i}, \; X_j(0) = x_j^{(0)}, \; (j = 1, 2, \ldots, n), \tag{10}$$

This expression can be written more compactly in the form of a matrix:

$$\frac{dx}{dt} = -\mu \nabla E(x), \; x(0) = x^{(0)}, \tag{11}$$

where $x(t) \in \mathbb{R}^n$, $\mu(x,t)$ is a positive definite matrix of dimension $n \times n$ whose inputs are generally dependent on time, and the variable $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$. Determining the positive matrix μ requires system stability. The next subsection describes a basic gradient system used to determine the direction of gradient change of a system of differential equations.

2.2. Basic Gradient System

Considering the simple case for which the matrix $\mu(x,t)$ is reduced to a positive scalar function $\mu(t)$, the system of differential equations according to (10) is simplified to:

$$\frac{dx_j}{dt} = -\mu(t) \frac{\partial E}{\partial x_j}, \tag{12}$$

with $x_j(0) = x_j^{(0)}$ $(j = 1, 2, \ldots, n)$; $\mu(t) > 0 \in C$ is a constant and theoretically it can be a large arbitrary set; the learning relationship $\eta^{(k)}$, in a discrete-time slope algorithm (ascending slope), is limited to a small interval to ensure that the algorithm converges. The previous system of differential equations is called the basic dynamic gradient system. Continuous-time gradient downward methods employ (12), since the search of $-\nabla E(x)$ is in the direction of maximum negative change of the objective function $E(x)$ at some point.

An interesting aspect of the method is the fact that the direction determined by the discrete-time iterative algorithm is slightly oscillatory. In contrast, the direction obtained by the continuous-time gradient method is monotonous. In other words, the oscillation effects can be eliminated using a continuous-time gradient system. As described, the main objective of this algorithm is to find a value of x that minimizes $E(x)$.

3. Analysis and Development

From the optimized model of the ANN, we carried out the CMOS circuit design based on the operational amplifier to design the base cell called perceptron, which is commonly used in multi-layer perceptron (MLP) architectures. Figure 1 shows the block diagram of the proposed methodology for the development of the circuit. Block 1 represents the numerical analysis stage and the simulation of the optimization of the objective function (obtaining synaptic weights and bias) for the ANN, based on gradient. Block 2 is focused on the design of the base neuron (perceptron) at the circuit level. Finally, block 3 shows the development and simulation of the complete MLP circuit to obtain the neural network behavior proposed in the case study.

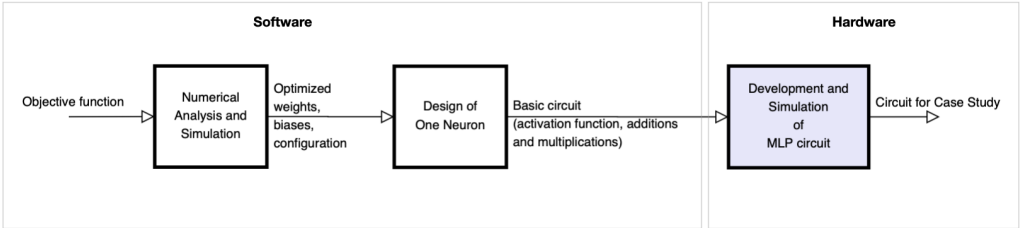


Figure 1. Block diagram of the design and development of the proposed methodology.

All the optimization algorithms described above employ a system of first-order differential equations. Now, we need to apply the optimization to a system of second-order differential equations. Therefore, to improve the convergence properties, we can use a system of higher-order ordinary differential equations by considering the system of second-order differential Equations (1), (3) and (8) as follows:

$$\delta(t)\frac{d^2x}{dt^2} = -\gamma(t)T\frac{dx}{dt} - \nabla E(x), \tag{13}$$

with initial conditions $x(0) = x^{(0)}$, $(\frac{dx}{dt})(0) = x'(0) = x'^{(0)}$, where $x(t) \in \mathbb{R}^n$, $\delta(t)$ and $\gamma(t)$ are positive functions with real values for $t \geq 0$, and $T = [T_{ij}]$ is a positive definite symmetric matrix of dimension $n \times n$.

A simple case is when δ, γ, and T are constant; for example: $\delta(t) = \delta_0 \geq 0$, $\gamma(t) = \gamma_0 > 0$, $T = [T_{ij}]$, and T_{ij} is constant for $i, j = 1, 2, \ldots, n$. However, in general, the matrix $T = T(x, t)$ depends on time. A particular choice of the parameters of (13) makes it possible to obtain almost all first-order methods. For example, when $\delta(t) = 0$, the conditions are the following:

1. For $T = I$ (identity matrix) and $\gamma(t) = \mu^{(-1)} > 0$, the gradient descent method is applied.
2. For $T = \nabla^2 E(x)$ (Hessian matrix) and $\gamma(t) = 1$, Newton's method is applied.
3. For $T = [\nabla^2 E(x) + v(t)I]$ and $\gamma(t) = 1$, the Levenberg–Marquardt method [11,12,15–17] is applied.

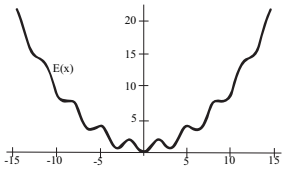
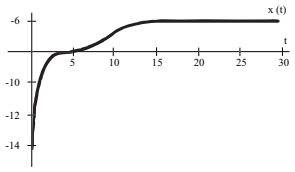
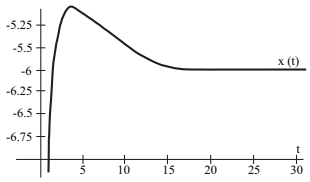
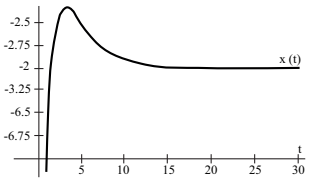
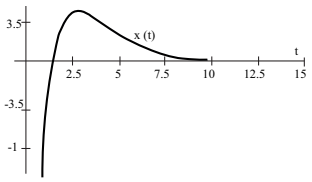
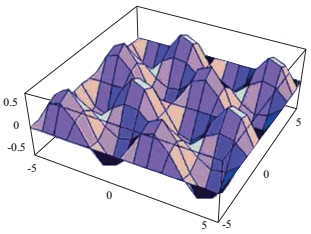
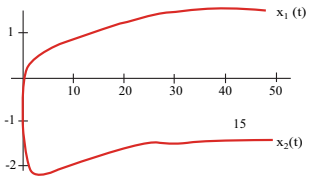
The system of second-order equations is inspired by classical mechanics and has the following physical interpretation [6]: (13) represents Newton's second law (mass $\times$ acceleration = force) for a mass particle $\delta(t)$ moving in a space $\mathbb{R}^n$ subject to a force $-\nabla E(x)$ given by the potential $E(x)$ and with force $-\gamma(t)T(\frac{dx}{dt})$. Since $\gamma(t) > 0$, the force $-\gamma(t)T(\frac{dx}{dt})$ is dissipative and $\gamma(t)$ is the coefficient of friction. Generally, the mass coefficient of $\delta(t)$ and the coefficient of friction $\gamma(t)$ are constant in time or tend to zero as time approaches infinity.

The application of a system of second-order differential equations has several important advantages over a system of first-order differential equations, which are:

1. Because of the initial force, the local minimum of the objective function $E(x)$ can be avoided by an appropriate choice of parameters, and the network can find an overall minimum, although this cannot be guaranteed.
2. The second-order differential equations have better flexibility. For example, for the same starting point $x(0)$ different from the selection of $(\frac{dx}{dt})(0)$, they can lead to a different local minimum. That is, we changed coefficients γ and δ making it possible to reach a local minimum from the same initial condition $(x(0), x'(0))$. Thus, in a system of differential equations of the form given by (13), an additional control of the solution is provided.
3. A system of second-order differential equations may have a better property of convergence. Therefore, responses can be obtained in its trajectory.

To evaluate the behavior of the gradient descent algorithm, we performed two simulations. In the first example, we have a single-variable objective function for which the determination of the local minimum depends on the initial condition of the variable. A two-variable objective function (which can fall into a saddle point or a local minimum) is described in the second example. This case can be avoided by an appropriate choice of the initial condition of the variables. The results obtained from the applications of the algorithm and the initial conditions are shown in Tables 1 and 2, respectively.

Table 1. Part One: Objective function.

Type of Function	Objective Function	Result
Minimizing the one-dimensional objective function: $E(x) = 2 * sin^2(x) + 0.1x^2, x \in \mathbb{R}$		 initial conditions $x(0) = -15$ and $x'(0) = 10$
 initial conditions $x(0) = -12$ and $x'(0) = 10$	 initial conditions $x(0) = -9$ and $x'(0) = 10$	 initial conditions $x(0) = -6$ and $x'(0) = 10$
Minimizing the n-dimensional objective function: E(x) = $sin(x_1) * sin(x_2), x \in \mathbb{R}^2$		

Applying (13), the optimization problem (one-dimensional) is:

$$\delta_0 \frac{d^2x}{dt^2} = -\gamma_0 \frac{dx}{dt} - \mu_0(4 * sin(x)cos(x) + 0.2x), \tag{14}$$

Applying (13), the optimization problem (n-dimensional) has the following form:

$$\delta_0(\frac{d^2x_1}{dt^2}) = -\gamma_0(\frac{dx_1}{dt}) - \mu_0 * cos(x_1) * sin(x_2), \tag{15}$$

$$\delta_0(\frac{d^2x_2}{dt^2}) = -\gamma_0(\frac{dx_2}{dt}) - \mu_0 * cos(x_2) * sin(x_1), \tag{16}$$

Table 2. Part Two: Initial Conditions of the Objective functions.

Type of Function	Initial Conditions of Objective Function
Minimizing the one-dimensional objective function	where μ_0 is a scaling factor. The coefficients δ, γ, and $T(= 1)$ are constant. For $\delta_0 = 0.1$, and $\gamma_0 = 0.15$, the initial conditions (starting points) are: $x(0) = -15$, -12, -9, -6, and $x'(0) = 10$.
Minimizing the n-dimensional objective function	with initial conditions $x_1(0) = -1$, $x_1'(0) = 2$, $x_2(0) = -1$, $x_2'(0) = -2$.

When a search of the local minimum of a particular objective function of two or more variables is performed, it is necessary to avoid falling into the saddle points caused by the initial conditions of the variables when calculating the solution. In Table 1, the results obtained for examples 1 and 2 show how we can improve the convergence properties of an

objective function with one or two variables. In general, an n-dimensional objective function will work, using the proposed system of higher-order ordinary differential equations. In the following section, we will show how this approach can be applied. We will do this through a case study for its application to an ANN implemented using analog systems and CMOS circuits.

4. Circuit Design: A Case Study of ANNs Implemented in CMOS Circuits

In this section, the application of the proposed system to an ANN [1,4,5] is described. An analog multiplier multiplies the outputs of a voltage adder concerning a constant. The voltage adder generates the sum of the voltages of the matrix W_{ij}, which represents the synaptic weight matrix (Figure 2), and the signal of the non-linear function generator. We used operational amplifiers (op-amps) to design each base neuron that composes the neural architecture. It means that we implemented the structure of the neural circuit of Figure 2 to produce the ANN represented in Figure 3. We remark that each neuron node has two parts: a sum function and an activation function (sigmoid). We implemented the first one with an op-amp inverting adder (designed based on Figure 4) and the second one with an array of op-amps and voltage limiters (diodes).

Figure 2 shows the block diagram for the function, which consists of two continuous-time integrators (whose response depends on the feedback network), one analog multiplier, one summing amplifier, and one non-linear function generator for calculating the gradient of the objective function at the circuit level. The optimized parameters $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ are the output signals of the integrator. This circuit is characterized by having a more robust output (insensitive to small perturbations) with respect to the parameter variation. The function generator is the only one that precisely calculates the gradient of the objective function.

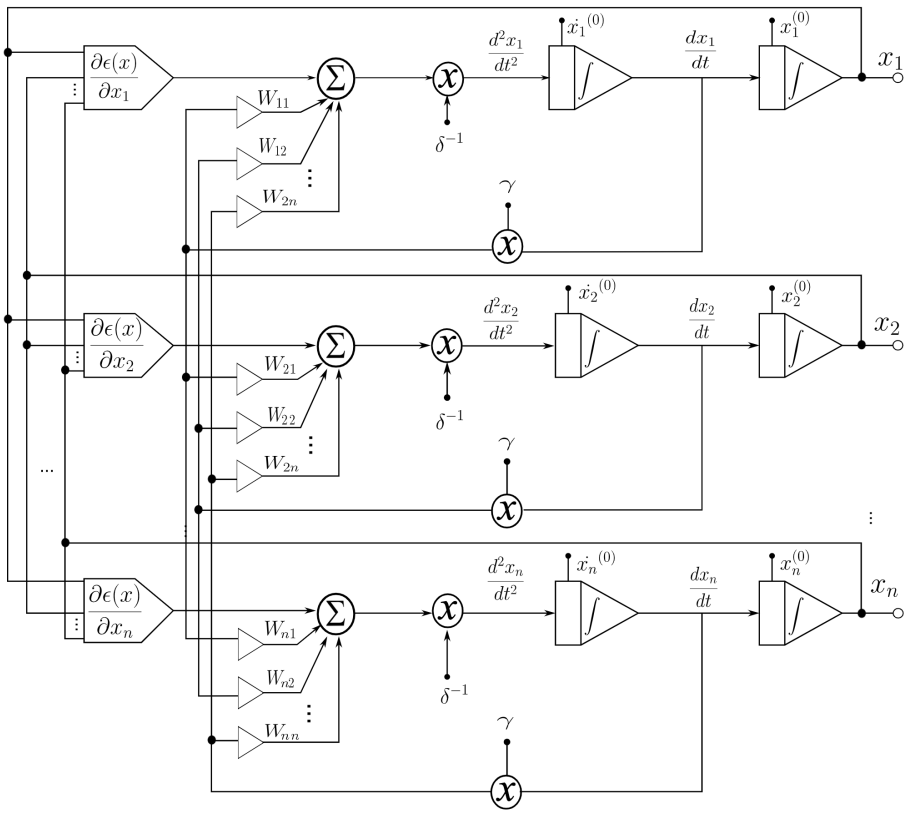


Figure 2. Block diagram for applications to ANNs.

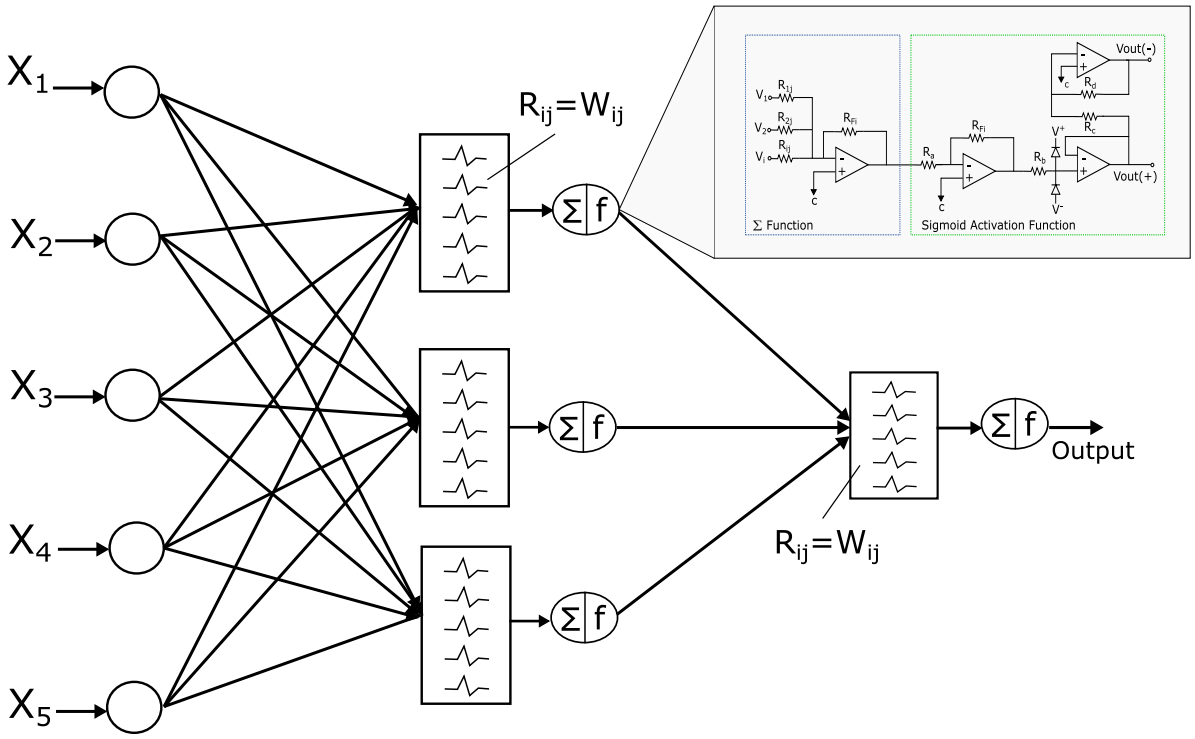


Figure 3. Complete circuit based on ANN.

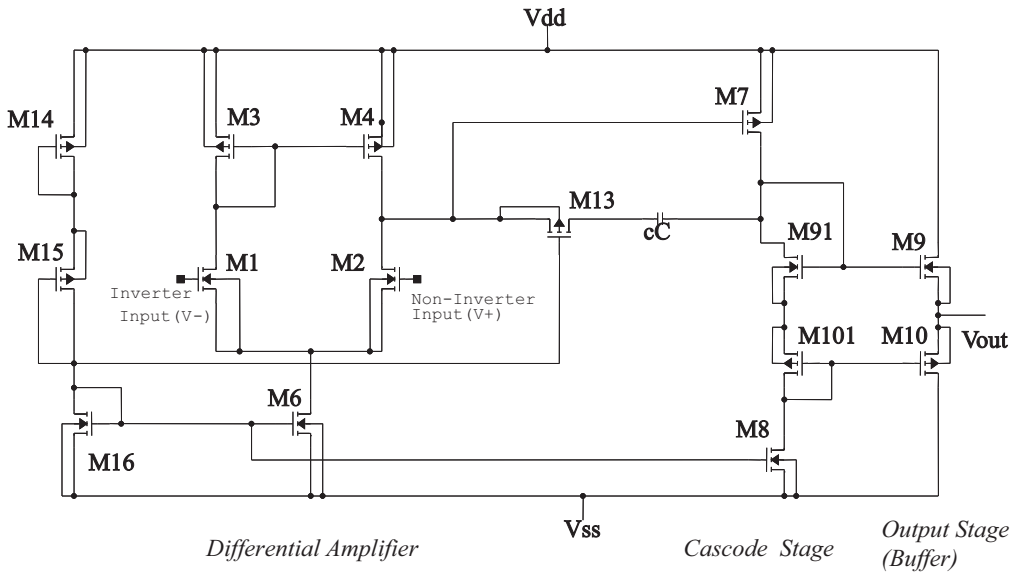


Figure 4. Operational amplifier 1.2 µm.

According to Figure 2, the design of the analog neural network consists of the development of the integrator circuit. As shown in Figure 5, the integrator circuit uses the op-amp shown in Figure 6 (the block diagram of the 1.2 µm technology operational amplifier in Figure 4). The op-amp is designed according to the specifications described in Table 3.

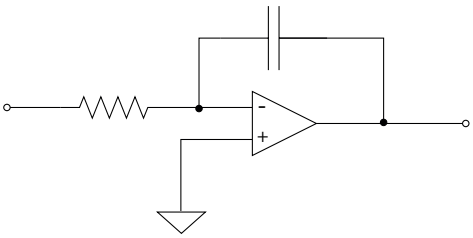


Figure 5. Integrator Circuit.

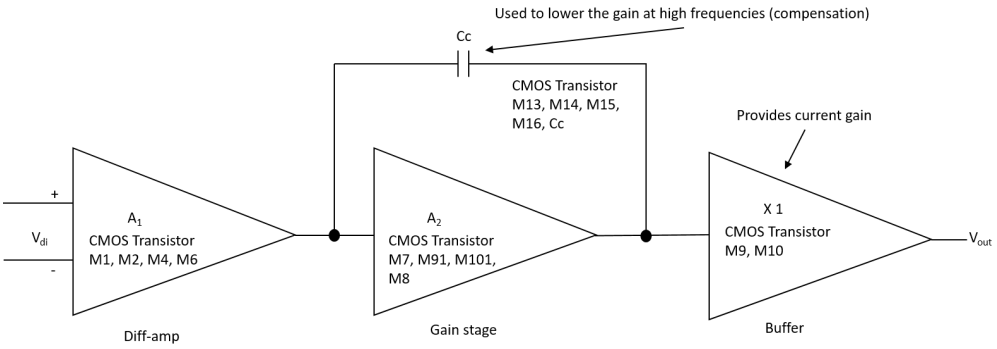


Figure 6. Block diagram of the 1.2 µm operational amplifier.

Table 3. Parameters of CMOS op-amp.

Parameter	Value
Open-loop voltage gain	60 dB
Phase margin (ϕ_o)	81°
Unity gain bandwidth (GB)	2.30 MHz
Slew rate (SR)	21.06 V/µs
CMRR	47 dB
Offset voltage (Vos)	284 µV
Compensation capacitance (CC)	8.75 pF
Capacitive load (CL)	20 pF
Power supply (VDD, VSS)	+/−2.5 V
Power consumption	1.28 mW
Common mode input voltage range	−608.696 mV to 1.98 V

Figure 6 is, in detail, the block diagram of the 1.2 µm technology operational amplifier in Figure 4 that shows the operational amplifier with CMOS transistors, formed by the stages of a differential amplifier (M1, M2, M4, and M6), gain stage (M7, M91, M101, and M8) and the output stage (buffer) (M9 and M10). Compensation network M13 and capacitor Cc, where M13 is made up of a polarization network by the transistors M14, M15, and M16. The dimensions of the main semiconductor N-channel and P-channel transistors are listed in Table 4. The characteristic values of the integrator are described in Table 5 [18,19].

The complete circuit of the analog neural network is shown in Figure 3, where the inputs X_i, the weights (defined by the resistors $R_{ij} = w_{ij}$), and the activation functions (Σ and F) are represented. The basic circuits that correspond to the analog neural network neurons are the inverting amplifiers and the activation function. The inverting amplifier circuit is used as a summation block. Hence, when many input voltages are connected to the inverting input terminal, the resulting output is the sum of all the input voltages applied, although inverted; this output, combined with the feedback resistor, generates the multiplication by a weight. The circuit for implementing a neuron is shown in Figure 7, where the function Σ computes multiplications and the activation function is a *Sigmoid* [19].

Table 4. Widths and lengths for N-channel and P-channel MOS transistors.

	2 µm Technology N-Channel MOS		**1.2 µm Technology P-Channel MOS**	
Transistor	**Width**	**Length**	**Width**	**Length**
		Differential Stage		
M1	15 µm	5 µm	9 µm	3 µm
M2	15 µm	5 µm	9 µm	3 µm
M3	70 µm	5 µm	42 µm	3 µm
M4	70 µm	5 µm	42 µm	3 µm
M6	30 µm	5 µm	18 µm	3 µm
M16	15 µm	5 µm	9 µm	3 µm
M14	70 µm	5 µm	42 µm	3 µm
M15	70 µm	5 µm	42 µm	3 µm
		Cascode Stage		
M13	70 µm	5 µm	42 µm	3 µm
M7	70 µm	5 µm	42 µm	3 µm
M91	15 µm	2 µm	9 µm	1.2 µm
M101	70 µm	2 µm	42 µm	1.2 µm
M8	15 µm	5 µm	9 µm	3 µm
		Output Stage		
M9	150 µm	2 µm	90 µm	1.2 µm
M10	700 µm	2 µm	420 µm	1.2 µm

Table 5. Characteristic Values of the CMOS integrator circuit.

Characteristic	**Values**
Slew rate	21.06 V/µs
Cutoff frequency	2.2570 MHz
Gain	60 dB
Phase Margin	83.782°
A1 gain	33.33
A2 gain	55.55

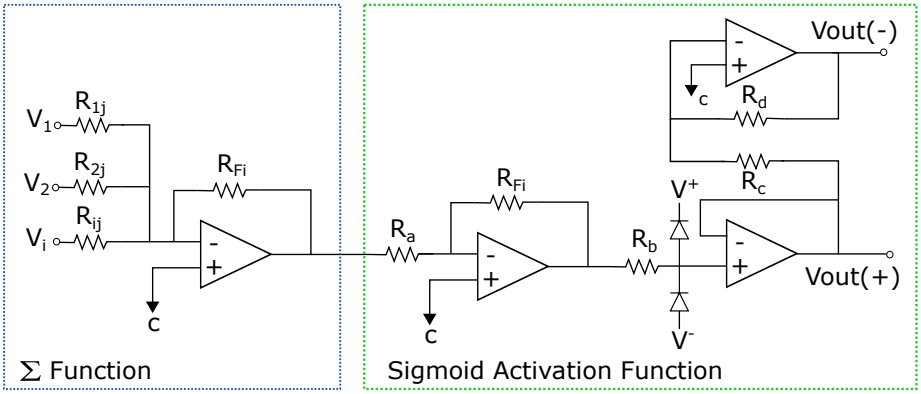


Figure 7. Circuit for one neuron.

As shown in Figure 3, the architecture has 9 neurons (configuration 5-3-1), and each neuron has 4 op-amps due to the adder and the sigmoid activation function (see Table 6). The total power consumption is 46.08 mW, because there are 36 op-amps for the entire circuit, and each of them consumes 1.28 mW [18,20,21].

Table 6. Number of op-amps and total power consumption.

Description	Quantity
No. of op amps per adder	1
No. of op amps per activation function	3
No. of op amps per neuron	4
No. of neurons for the 5-3-1 architecture	9
No. of op amps for the 5-3-1 ANN circuit	36
Total power consumption	46.08 mW

5. Application to an Implemented Analog ANN for Pattern Detection

The ANN-based architecture of this work (see Figure 3) is proposed for pattern recognition as an application example, where the recognition task consists of detecting a value of +1 or −1 for Hamming code correction. One of the advantages of the proposed circuit is that it can be integrated as a module in some communication systems in application-specific integrated circuits (ASIC).

The electronic simulation of the backpropagation neural network is implemented on the basic circuits designed for a typical neuron (the weighted sum function and the activation function). This simulation is carried out using the Pspice program [18], which allows changing the device parameters as well as the stimuli, in such a way that different operating conditions of the circuit are covered.

Once modeled the behavior of the cells that make up the backpropagation, the complete circuit simulation to study the network performance as a whole can continue. This is done by connecting the synapses and the activation functions according to the configuration established in Figure 3. The input vector is bipolar (see Table 7).

Figure 8 represents the neural network response (designed to classify input patterns with a supervised learning algorithm) to different stimuli, with a sweep of the output signal concerning the input signal. Figure 8a represents a stimulus P = [−1 1 1 1 1] and the corresponding output T = [1]; if we observe from left to right, we have a negative input and a positive posterior one, which is consistent with the input pattern and its positive output: 1. Figure 8b shows the behavior of the architecture with a stimulus P = [−1 −1 −1 −1 −1] and an output T = [1]; if we sweep the input signal from left to right, we can observe that all the input patterns are negative and we have a positive output at 1. For Figure 8c we have an input pattern P = [1 −1 −1 −1 −1] and an output T = [−1], sweeping from left to right the input pattern is negative and then positive; where the first position of the input vector will be the least significant one for the response of the architecture, resulting in a negative output. Finally, for Figure 8d we have an input pattern P = [1 −1 1 1 1] and an output T = [−1]; if we sweep the output signal from left to right, we observe that the negative value of the pattern is not preponderant for the output of the architecture, which in this case is negative.

The proposed methodology is based on the problem analysis, the datasets definition (inputs and outputs) for a supervised method, the evaluation of ANN configurations, the selection of the best configuration, the implementation of the configuration through operational amplifiers, and the simulation of the ANN-based circuit. In this work, a detection circuit is implemented. The outline training is done through the use of Mathematica (i.e., the training, in which weights and biases are estimated, is a software-based step carried out prior to the execution of the simulation of the ANN-based circuit), and the ANN configuration is carried out using op-amps.

Table 7. Bipolar input vector.

Input Vector (P)					Target (T)
−1	−1	−1	−1	−1	1
−1	−1	−1	−1	1	1
−1	−1	−1	1	−1	1
−1	−1	−1	1	1	1
−1	−1	1	−1	−1	1
−1	−1	1	−1	1	1
−1	−1	1	1	−1	1
−1	−1	1	1	1	1
−1	1	−1	−1	−1	1
−1	1	−1	−1	1	1
−1	1	−1	1	−1	1
−1	1	−1	1	1	1
−1	1	1	−1	−1	1
−1	1	1	−1	1	1
−1	1	1	1	−1	1
−1	1	1	1	1	1
1	−1	−1	−1	−1	−1
1	−1	−1	−1	1	−1
1	−1	−1	1	−1	−1
1	−1	−1	1	1	−1
1	−1	1	−1	−1	−1
1	−1	1	−1	1	−1
1	−1	1	1	−1	−1
1	−1	1	1	1	−1
1	1	−1	−1	−1	−1
1	1	−1	−1	1	−1
1	1	−1	1	−1	−1
1	1	−1	1	1	−1
1	1	1	−1	−1	−1
1	1	1	−1	1	−1
1	1	1	1	−1	−1
1	1	1	1	1	−1

In [22,23], the authors presented a circuit-level implementation of the backpropagation learning algorithm, exploring the gradient descent method in the design of analog circuits. They used an ANN in memristive crossbar arrays, simulating in SPICE on TSMC's 180 nm CMOS technology, reporting output voltages and timing behaviors.

In our case, we presente an analog system simulation based on the solution to optimization problems with objective functions of one and two variables using the gradient descent method (ascending slope). We showed the ANN configuration behavior (learning algorithm), where its input and process signals are analog. Our circuit-level implementation was made with a 5-3-1 configuration using the backpropagation training algorithm, reporting the simulation-level architecture results in PSpice level 9.1 on 1.2 µm technology with a lambda of 0.5 µm, with widths and lengths of the X-channel defined by the technology.

Although the learning of the neural network was developed offline, and we consider the optimization results obtained in the mathematical part of the article were satisfactory, we emphasize that its essential purpose was to illustrate the applicability to optimize the learning algorithm of an ANN. We also simulated the optimized ANN model through the design of a circuit with CMOS transistors.

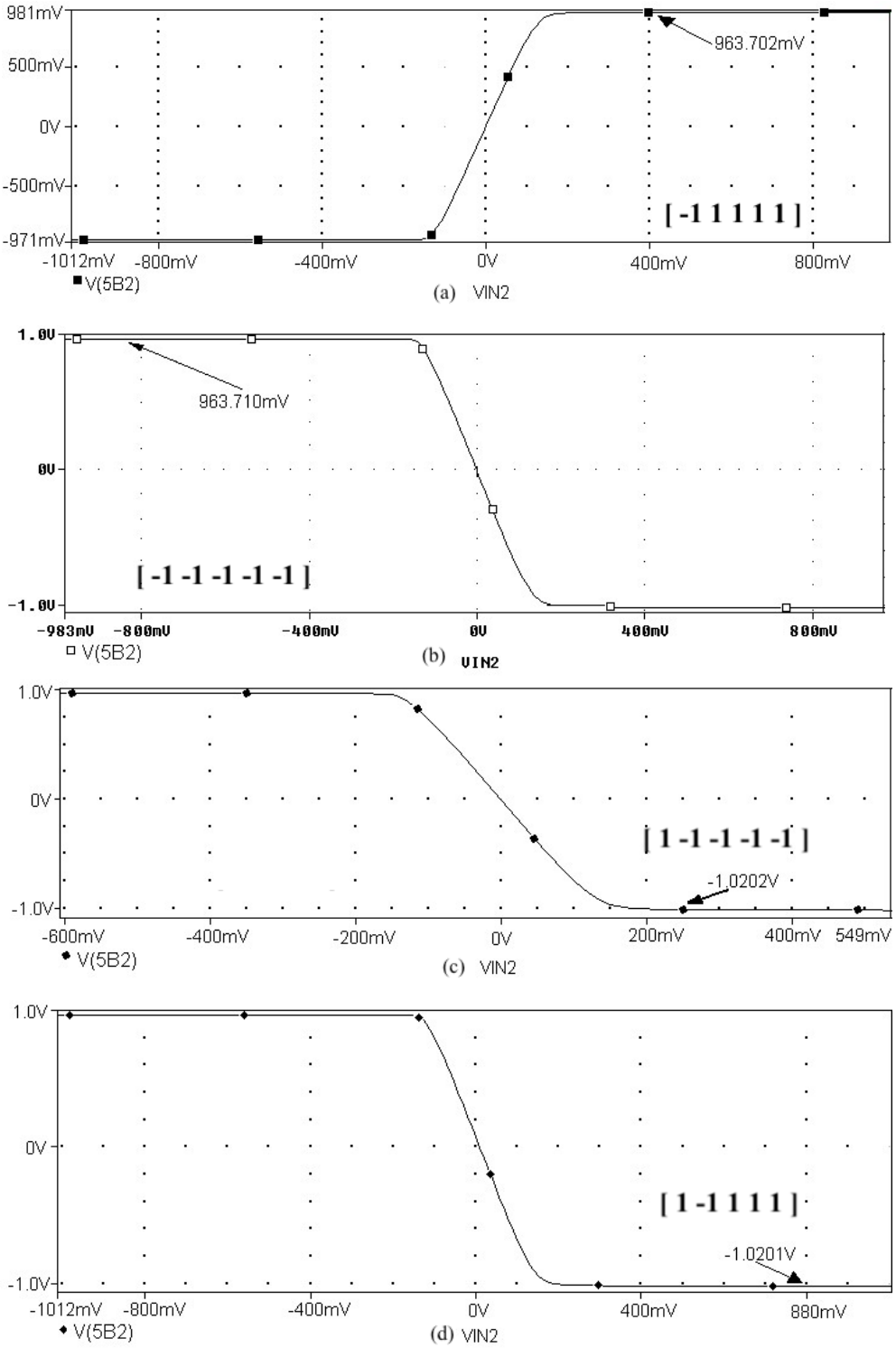


Figure 8. Response of the neural network to different stimulus. The stimulus means external inputs to the network. (**a**) P = [−1 1 1 1 1]; T = [1], (**b**) P = [−1 −1 −1 −1 −1]; T = [1], (**c**) P = [1 −1 −1 −1 −1]; T = [−1], and (**d**) P = [1 −1 1 1 1]; T = [−1].

6. Comparison between Our Proposal and Related Works

Hardware design for neural architectures using base neurons built with CMOS transistors is a fundamental part of our design in comparison with the current cited works. Although we cannot compete in processing speed, we have the advantage of implementing ANN circuits with standard cells without modifying circuits for implementation. We

chose several aspects to carry out a comparison among similar works [21–23] (see Table 8) considering: (1) the chosen material or device type, (2) the kind of device on which the synapsis is based, (3) the amount of power required, and (4) the learning algorithm used.

Table 8. Comparison between our proposal and related works.

Work	Device Quirks	Synapsis Based on	Power Consumption (mW)	Learning Algorithm
Our Proposal	CMOS	op-amps	46.08	BP
Zhang et al. [21]	Memristor	Memristor array + op-amps	-	BP
Krestinskaya et al. [22]	Memristive Crossbar	CMOS	115.65	BP
J. Han, et al. [23]	FPGA	LUT, FF, BRAM	477	SNNs

The synaptic weights of the proposed method, presented in Table 3, were obtained offline by the Backpropagation learning method for the analog implementations and by a Spiking Neural Network (SNN) for the field-programmable gate array (FPGA) implementation. In our proposal, the synaptic weights were implemented with resistors and op-amp circuits to form the base cell. In reference [21] the synapses were implemented with an array of resistances and op-amps. In contrast, in reference [22] the synapses were implemented by means of cells formed in the Memristive Crossbar. Finally, in the FPGA [23] they were implemented through modules that perform fixed arithmetic calculations using the hardware resources of the family.

A comparison between the proposed work and the implementation using FPGAs is somehow unfair because of the density of programmable logic of the latter, considering energy consumption, processing speed, and clock cycles, among other factors.

Power consumption is directly proportional to the density of the devices used to design the architectures at a hardware level. In the case of our proposal, power consumption is due just to op-amps and resistances. In the case of [22], power consumption is subject to the cells of the transistors, where the effect of signal propagation delay is not negligible. Finally, in the case of the FPGA implementation [23], power consumption was because of the density of programmable logic gates for the whole system; where we should consider static and dynamic powers. For instance, there is a device with unused hardware resources though consuming power since hardly an architecture will consume 100% of the FPGA resources, generating a non-optimal consumption of resources.

The architecture of [23] and our proposal are very different, in our case the architecture presents 9 neurons implemented in parallel and in the case of [23], the authors reported an architecture that calculated the 2842 neurons by using an iterative process (processor-type architecture), where a single neuron was implemented, and this implementation was able to compute the result of all neurons due to the design of the proposal. In this way, if the architecture of [23] implemented 9 neurons, as the same basic architecture requires, it would be consuming similar energy as they reported, since it was a processor architecture that was operating. Implementing 2842 neurons in the FPGA with floating numbers and their processing modules for addition, subtraction, etc., would require greater power consumption and hardware resources, which can be limited by the amount of FPGA resources.

We cannot compare the work in [22] and ours directly, even though both presented simulations carried out in Matlab and Pspice, respectively. For both cases, the technology and implementation were different. However, it is reasonable to think that the better the technology, such as memristors, the lower the energy consumption. In various works, such as in [24], it was said that the energy consumption of [22] was high, and they seeked to lower it.

The work in [22] was developed in 180 nm TSMC's CMOS technology. The implementation and simulation of the circuit were carried out with memristors and in a crossbar

configuration. The authors of [22] implemented a neural network with three neurons in the input layer, two neurons in the output layer, and five neurons in the hidden layer. In contrast, we proposed an ANN made with CMOS with 1.2 µm technology with five neurons in the input layer, one neuron in the output layer, and three neurons in the hidden layer.

Power dissipation was lower in our case because in the CMOS design, being this a mature technology, there are rules that must be satisfied [25–27]. On the other hand, in the case of memristors, their implementation in the commercial area is not available. Hence, the silicon organization of their components is still being built or defined. In the case of speed and performance, it is not competitive compared to DRAM, which is used by CMOS technology.

Other factors can affect the behavior of the circuits, but since they are different technologies, their usefulness in the comparison is null, so they are not part of the analysis of this work, such as frequency effects, transistor channel dimensions (W/L), channel resistance, channel conductivity, voltage threshold, response speed, scaling of CMOS devices, reduced geometry effects, channel length, and CMOS channel width.

7. Conclusions

In this work, an MLP architecture design at the hardware level was presented, using base neurons at the circuit level as its building blocks. Neuron weights and bias were optimized off-line using n-dimensional objective functions through the gradient descent method. The hardware architecture of the base neuron was designed on a CMOS operational amplifier in 1.2-micron technology. The total power consumption was 46 mW. The 5-3-1 ANN architecture was designed with nine neurons, using 36 op-amps, demonstrating its electronic performance in the practical case shown in Section 5.

The application developed with the implementation of the base neuron circuits showed optimal results in detecting the pattern. Starting with the supervised learning algorithm and based on the gradient descent method to obtain the synaptic weights and bias for the neural architecture, the responses shown in Figure 8 and Table 7 were appropriate to the electrical behavior. The MLP architecture presented an efficiency of 99 %. We might suggest other applications, but in the first instance, we determined that solving this problem was an adequate choice to show the efficiency of the MLP system. Other applications could be proposed, but that remains as future work; for example, we are currently working on image recognition to classify agricultural products and on data processing of hypertensive and diabetic patients. As part of the future work, we have the design and implementation of neural network circuits based on the optimization of objective functions. Besides, we can use other tools, such as Verilog-AMS or VHDL-AMS, as alternatives to simulate the results presented in this work. Another future work is trying other meta-heuristic optimization methods to obtain better ANN training results. We consider that the application of the proposed method was optimal but not without leaving a possibility of applying another meta-heuristic method in the short term.

The advantage of our development with respect to other works cited in Table 8, can be observed, mainly, in the following improvements: power dissipation, number of transistors, and probably, the dimension in area of the integrated circuit. It is likely that one of the disadvantages of our implementation is the speed of response, but that is compensated by the fact that this is a design containing just what is necessary for the application, without an excessive amount of resources.

Author Contributions: Conceptualization, A.M.-S., C.A.H.-G., L.A.M.-R. and I.A.-B.; methodology, A.M.-S., C.A.H.-G., L.A.M.-R. and I.A.-B.; software, A.M.-S., M.A.G. and J.A.O.T.; validation, A.M.-S., I.A.-B. and M.A.G.; formal analysis, A.M.-S. and L.A.M.-R.; investigation, A.M.-S., C.A.H.-G., L.A.M.-R. and I.A.-B.; resources, A.M.-S., C.A.H.G., L.A.M.-R., I.A.-B., M.A.G. and J.A.O.T.; writing—original draft preparation, A.M.-S., I.A.-B., L.A.M.-R. and C.A.H.-G.; writing—review and editing, A.M.-S., C.A.H.-G., L.A.M.-R. and I.A.-B.; visualization, C.A.H.-G. and J.A.O.T.; supervision, L.A.M.-R. and A.M.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Mexican National Council for Science and Technology (CONACYT) through the Research Projects 882, 278, and 613.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the study's design, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Baker, R.J. *CMOS: Circuit Design, Layout, and Simulation*, 4th ed.; Wiley-IEEE Press: Hoboken, NJ, USA, 2019.
2. Razavi, B. *Design of Analog CMOS Integrated Circuits*, 2nd ed.; McGraw-Hill Education: New York, NY, USA, 2016; ISBN 0072524936.
3. Abbas, K. *Handbook of Digital CMOS Technology, Circuits, and Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; ISBN 9783030371944.
4. Mandal, R.K. Design of a CMOS OR Gate using Artificial Neural Networks (ANNs). *AMSE J. Ser. Adv. D* **2016**, *21*, 66–77.
5. Miona, A.; Vančo, L. Application of Artificial Neural Networks in Electronics. *Electronics* **2017**, *21*, 87–94. [CrossRef]
6. Cichocki, A.; Unbehauen, R. *Neural Networks for Optimization and Signal Processing*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1993.
7. Rizaner, F.B.; Rizaner, A. Approximate Solutions of Initial Value Problems for Ordinary Differential Equations Using Radial Basis Function Networks. *Neural Process. Lett.* **2018**, *48*, 1063–1071. [CrossRef]
8. Tan, L.S.; Zainuddin, Z.; Ong, P. Solving ordinary differential equations using neural networks. *AIP Conf. Proc.* **2018**, *1974*, 020070.
9. Pates, R.; Vinnicombe, G. Scalable Design of Heterogeneous Networks. *IEEE Trans. Autom. Control* **2017**, *62*, 2318–2333. [CrossRef]
10. Haug, Y.; Guo, N.; Seok, M.; Tsividis, Y.; Mandli, K.; Sethumadhavan, S. Hybrid Analog-Digital Solution of Nonlinear Partial Differential Equations. In Proceedings of the 2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Cambridge, MA, USA, 14–18 October 2017. [CrossRef]
11. Hagan, M.T.; Demuth, H.B.; Beale, M.H. *Neural Network Design*, 2nd ed.; PWS Publishing Company: Boston, MA, USA, 2014.
12. Adolphs, L.; Daneshmand, H.; Lucchi, A.; Hofmann, T. Local Saddle Point Optimization: A Curvature Exploitation Approach. Proceedings of Machine Learning Research. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, Naha, Japan, 16–18 April 2019.
13. Rhinehart, R.R. *Engineering Optimization: Applications, Methods and Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 2018.
14. French, M. *Fundamentals of Optimization Methods, Minimum Principles and Applications for Making Things Better*; Springer: Berlin/Heidelberg, Germany, 2018.
15. Bogdan, M.; Wilamowski, J.; David, I. *Intelligent Systems*; CRC Press: Boca Raton, FL, USA, 2018. [CrossRef]
16. Li, Y. *Deep Reinforcement Learning: An Overview, Neural Networks*; Elsevier: Amsterdam, The Netherlands, 2018.
17. Markopoulos, A.P.; Georgiopoulos, S.; Manolakos, D.E. On the use of back propagation and radial basis function neural networks in surface roughness prediction. *J. Ind. Eng. Int.* **2016**, *12*, 389–400. [CrossRef]
18. Lai, J.L.; Wu, C.Y. Design Ratio-Memory Cellular Neural Network (RMCNN) in CMOS Circuit Used in Association-Memory Applications for 0.25 mm Silicon Technology. *Open Mater. Sci. J.* **2016**, *10*, 54–69. [CrossRef]
19. Su, Z.; Kolbusz, J.; Wilamowski, B.M. Linearization of Bipolar Amplifier Based on Neural-Network Training Algorithm. *IEEE Trans. Ind. Electron.* **2016**, *63*, 3737–3744. [CrossRef]
20. Burr, G.W.; Shelby, R.M.; Sebastian, A.; Kim, S.; Kim, S.; Sidler, S.; Virwani, K.; Ishii, M.; Narayanan, P.; Fumarola, A.; et al. Neuromorphic computing using non-volatile memory. *Adv. Phys. X* **2017**, *2*, 89–124. [CrossRef]
21. Zhang, Y.; Wang, X.; Friedman, E.G. Memristor-Based Circuit Design for Multilayer Neural Networks. *IEEE Trans. Circuits Syst. Regul. Pap.* **2018**, *65*, 677–686. [CrossRef]
22. Krestinskaya, O.; Salama, K.N.; James, A.P. Analog Backpropagation Learning Circuits for Memristive Crossbar Neural Networks. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
23. Han, K.J.; Li, Z.; Zheng, W.; Zhang, Y. Hardware Implementation of Spiking Neural Networks on FPGA. *Tsinghua Sci. Technol.* **2020**, *25*, 479–486. [CrossRef]
24. Nafea, S.F.; Dessouki, A.A.S.; El-Rabaie, S. Summary of memristor up to 2015. *Menoufia J. Electron. Eng. Res.* **2015**, *24*, 79–106. [CrossRef]
25. Wang, Y.; Cao, S.; Jin, X.; Peng, Y.; Luo, J. *Semiconductor Science and Technology*; IOP Publishing Ltd.: Bristol, UK, 2020; Volume 36.
26. Yener, Ş.; Kuntman, H. New CMOS based memristor implementation. In Proceedings of the 2012 International Conference on Applied Electronics, Pilsen, Czech Republic, 5–7 September 2012.
27. Abunahla, H.; Mohammad, B. *Memristor Technology: Synthesis and Modeling for Sensing and Security Applications, Chapter 1 Memristor Device Overview and Chapter 6 Memristor Device Modeling*; Springer: Berlin/Heidelberg, Germany, 2018.