Analog RF Circuit Sizing by a Cascade of Shallow Neural Networks

Philippe-Olivier Beaulieu, Étienne Dumesnil, Frederic Nabki, and Mounir Boukadoum
ÉTS, Montreal, Canada; UQAM, Montreal, Canada

***Abstract*—A deep neural network architecture for the automatic sizing of analog circuit components is proposed, with a focus on radio frequency (RF) applications in the 2 to 5 GHz region. It addresses the challenges of the typically small number of examples for network training and the existence of multiple solutions, of which impractical values for integrated circuit implementation. We address these issues by restricting the learning to one component size at a time, thanks to a cascade of dedicated shallow neural networks (SNN), where each network constrains the prediction of the next ones. Moreover, the SNNs are individually tuned by a genetic algorithm for the prediction order and accuracy. This reduction of the solution space at each step allows the use of small training sets, and the introduced constraints between SNNs handle component interdependencies. The method is successfully validated on three different types of RF microcircuits: a low-noise amplifier (LNA), a voltage-controlled oscillator (VCO), and a mixer, using 180 nm and 130 nm CMOS implementations. All the predictions were within 5 % of the true values, both at the component and performance levels, and all the responses were obtained in less than 5 s, after 4 to 47 min. training on a regular PC station. The obtained results show that the proposed method is fast and applicable to arbitrary analog circuit topologies, with no need to retrain the developed neural network for each new set of desired circuit performances.**

Index Terms— Design automation, circuit sizing, microelectronics, analog circuits, radiofrequency, RF, deep learning, neural networks, genetic algorithms.

I. Introduction

Currently, the typical design flow of analog circuits still requires substantial human intervention [1], since designer experience must compensate for properties and implementation factors not accounted for by the existing design tools, including non-linear components, bias requirements, and real-world effects like stray impedances, physical circuit layout, and component coupling. This usually results in a time-consuming iterative design process that must be repeated for each new design, and the situation is likely to worsen with the increasing complexity of electronic circuits and systems, and the shortening of their useful life spans.

Many works have proposed circuit design workflows for electronic design automation (EDA), particularly for digital integrated circuits and systems. However, the design and synthesis/sizing of analog circuits, particularly for radiofrequency (RF) applications, are still challenging due to the aforementioned limitations.

Circuit design is usually more complicated than analysis. In the former, the tools uniquely determine a circuit's behavior from its topology and component values, but circuit design addresses the reverse problem: find a feasible circuit topology and component values to match given performances. Formally, two problems are reversible with respect to each other if the formulation of one involves all or part of the solution of the other [2]. Then, the better-understood problem is said to be direct, while the other one is said to be inverse. For, analog circuits, analyzing circuit behavior given a topology and component values is a direct problem, while synthesizing the circuit (i.e. finding the topology and component values) for a given behavior is an inverse problem, and it usually affords multiple solutions, not all feasible in practice. This includes outsized geometries or component values.

Automating the solution of inverse problems has been a focus of research in many fields since they arise whenever a physical system is to be inferred from property measurements [3], and their well-posedness must be established for a straightforward solution to exist. Formally, a problem must meet three criteria to be well posed [4]: 1) it can be solved; 2) the solution is unique; 3) the solution is continuous with respect to data and parameter changes. A problem that does not respect those criteria is ill-posed, and this is often the case for inverse problems as they usually afford multiple solutions, and they may also violate the third criterion, as small data or parameter changes in them may lead to wide variations in the output values or solution accuracy.

The inverse problem of circuit design can be simplified if the circuit topology is given, since the design scope then reduces to estimating the component sizes. However, it is difficult to model RF circuits by regular linear equations with lumped parameters at GHz frequencies, since electric and magnetic fields must be accounted for, along with dissipative losses [5], leading to complex wave equations, and secondary effects such as those of stray capacitances and layout effects arise. In this context, the typical approach follows a lengthy iterative process of simulation and analysis, followed by sizing adjustments, and an efficient automatic sizing solver can be of great assistance. There also exist commercial products such as Neolinear [6], Solido [7], and MunEDA [8] that have been developed to help, but their lack of genericness across circuits and technologies, and the need to reconfigure them for each new design have been obstacles to a wide adoption. In practice, the choice of topology and the sizing of components are application-specific, making it harder for one algorithm to perform well in all situations. In addition, the set-up and configuration costs are important since the designers must use different tools and design environments. Finally, because of the high number of circuit topologies, technologies, and performance metrics, there are no specific benchmarks to evaluate and compare the available EDA algorithms [9].

Analog RF circuit design is typically a mix of methods and experience where, after selecting a circuit topology, a component sizing process takes place, followed by drawing the circuit layout and extracting the parasitic components and secondary effects to fine-tune it. In this paper, a neural network-based methodology is proposed to speed up the initial sizing step by learning from a relatively small set of solved examples. Then, given an RF circuit and a set of desired performances, it automatically sizes the circuit components

before the tuning step, to within 5% of both the required values as determined by the usual simulations and the desired performances. The approach is fast and generic for any circuit topology or performance specifications, and our validation results show that it predicts the sought component values within the set accuracy and performance thresholds, without being affected by the implementation geometry.

The proposed approach consists in a cascade of progressively built shallow neural networks (C-SNN), where each SNN is individually specified by a genetic algorithm (GA) to predict one component size. Moreover, each SNN output constrains the subsequent SNNs for mutual compatibility. This one-by-one approach simplifies the search space to allow for smaller training sets while forcing the generated component sizes to be compatible.

We define a shallow neural network as one that includes two hidden layers at most. Here, it consists of a multilayer perceptron (MLP) whose hyperparameters are tuned by a GA. Then, each trained MLP adds its output to the desired performances to constrain the learning of the next MLPs in the cascade while compensating for potential coupling with the already predicted values. The prediction order is also determined by the GA.

Three specific analog RF circuits are used for validation, with one of them implemented in 180 nm CMOS technology and the other two in 130 nm CMOS technology, but the proposed methodology is circuit and technology-agnostic, and other choices could have been made by simply using the appropriate set of completed designed as training examples. The three circuits were selected mainly to represent basic building block functionalities in RF circuits. As the proposed tool only accomplishes the initial sizing step of the design cycle as implemented, the used training data do not account for the effects of parasitic components or electromagnetic (EM) interference. This will be discussed further in Section VI.

The balance of this paper is as follows. Section II reviews the related work on component sizing; Section III presents the C-SNN architecture, along with the GA used to optimize its hyperparameters; Section IV describes the RF microcircuits used for validating the proposed method, with Section V providing the obtained results; finally, Section VI offers a discussion and concluding remarks.

II. Related Work

Initially, two major approaches could be identified in relation to analog design automation, knowledge-based and metaheuristic-based. Today, artificial neural networks (ANNs) constitute a promising third alternative. Below is a brief survey of previous work devoted to the analog circuit sizing problem, with a justification of the present work at the end.

A. Knowledge-Based Methods

These techniques are among the oldest and strive to imitate the behavior of expert designers [10], like plans to set the values of the design parameters in steps [11]. The approach is effective at low frequencies and for relatively small circuits such as operational amplifiers [12]. However, larger circuits rapidly increase in complexity, with non-linear relations and couplings to account for between the components and the circuit behavior, making it difficult to build an efficient design plan [13][11]. FEATS [14] is a method that uses abstract building blocks to create and evaluate topologies based on known circuits. This methodology is flexible in terms of circuit classes, technology nodes, and performance measurements, but generates many meaningless interim structures. A similar open-source design methodology based on the functional block is FUBOCO [15]. A library for different functional stages (bias, load, differential pair, etc.) is used in conjunction with composition rules to create and evaluate the topologies. The methodology is more complicated than FEATS but reduces the search space for the desired circuit topology. In all cases, a good knowledge of the design rules is required to use the previous methodologies. Moreover, as knowledge-based methods are essentially expert systems, they suffer from the fundamental difficulty to extract human expert knowledge (i.e. rendering explicit an essentially implicit procedural knowledge) [16]. This has led many researchers to look for methods that can autonomously learn to find solutions.

Case-based reasoning (CBR) [17][18] attempts to circumvent the problem by shifting the focus on the expert thinking's outcome instead of the thinking itself. CBR can be summarized as the search and retrieval of a "close" solved design and its revision for adaptation to the current case. If the retrieved circuit behavior is the same as the required one, its component values are just reused; if not, knowledge-based methods are used to revise the component parameters [19]. However, this requires many stored designs to be efficient [12], which makes its usefulness limited for circuit design, synthesis, or sizing, as only a few examples are usually available.

B. Metaheuristics

Metaheuristic-based methods view the sizing problem as multi-objective constrained optimization [20][13][21], and they typically find the solution by using a search algorithm.

1) Simulated Annealing

Simulated annealing [22] explores the solution space with increasingly finer granularity until a final solution is reached (e.g. see [23][24]). The approach has two important limitations in the context of circuit sizing. First, the circuit of interest must be simulated each time for error evaluation, making the algorithm time-consuming for complex problems. Second and more importantly, the final sizing solution is specific to the analyzed circuit. Indeed, the method does not learn "how to" size, but only finds the idiosyncratic solution for a single set of required circuit behavior. Thus, the optimization algorithm must be restarted for each new desired behavior. This can be a serious limitation in terms of sizing time. For example, the simulated annealing method proposed in [19] needed between one and three hours on a 2.3 GHz CPU to complete the sizing of each low-noise amplifier (LNA) performance criteria it was given. The one proposed in [23] required approximately 25 seconds to size a new operational amplifier and one hour to size a new voltage-controlled oscillator (VCO) on a 2.4 GHz CPU.

2) Genetic Algorithm

GA search is similar to simulated annealing, in that a competitive process is used to find a solution (e.g. see [25][26]). However, instead of two competing neighbors at each iteration, a whole population of potential solutions is involved, each one using a "chromosome" metaphor. In the context of circuit sizing, the chromosomes typically define the circuit component values and their population goes through a series of selection-reproduction-mutation cycles until convergence towards a solution [27]. Like simulated annealing, the returned component values apply to a specific set of performance

criteria, and the process can be time-consuming due to repetitive circuit simulation. For example, the GA implemented in [28] took approximately 54 seconds on a 2.8 GHz CPU to complete the sizing of an operational amplifier, while one implemented in [25] took consistently more than 30 minutes to size a LNA. In both cases, the process had to be repeated for any new set of performance criteria presented. Generic algorithms are also used to optimize other approaches [29].

A variant of GA, differential evolution, uses real-valued vectors as genotypes and the difference between vectors in breeding new generations. In [30], it is combined with surrogate models of electromagnetic simulations (EM) for an end-to-end design. However, the approach must be restarted for each circuit instance and its time consumption for the three provided examples varied between 42 and 106 hours on an average desktop station.

3) Genetic Programming

Genetic programming (GP) attempts to solve the synthesis and sizing problems using a dynamically bred program [31][32]. It has been shown to perform well not only to find an adequate set of values for components but also to find an adequate topology, including selecting the actual components and interconnections of the circuit. As proposed in [33], a tree is used to represent the analog microelectronic circuit. First, an embryonic circuit is generated, from which the final circuit is evolved. The evolution of the circuit reflects the evolution of the functions that constitute the branches of the circuit-constructing program tree (see also [34] for a different coding scheme). The main advantage of GP over GA and simulated annealing is its efficiency at synthesizing the whole circuit instead of being limited to circuit sizing. Many variations of the genetic programming algorithm have also been proposed for circuit sizing (e.g. [34]), reaching efficient sizing optimizations. However, in the context of the present work, they also have the limitations of the preceding metaheuristics.

In all the previous approaches, a design-in-the-loop approach is used, in which the circuit of interest is built or simulated for testing during the algorithm iteration process. The approach can result in very long convergence times [30][35], even with approximation techniques such as surrogate circuits are used for simulation, and final tuning is done by an expert [36].

The preceding subsections are just examples of the various techniques used to tackle the optimization problem with metaheuristics, and many other approaches have been reported in the literature. For example, [37] used Bayesian optimization for the design of RF power amplifiers, and [38] used Bayesian Model Fusion to reuse early-stage data when fitting a late-stage performance model of two circuits mixed signal circuits.

C. Artificial neural networks

Neural networks differ from knowledge-based techniques and metaheuristics by viewing the sizing problem as one of classification/regression. Using a set of successful circuit design examples, they use their generalization ability to specify the component values of similar new circuits. This capability makes them more generic than the previous methods which must be started over for each new design, even when the same circuit topology is used. As the name implies, the approach relies on artificial neural networks (ANN), especially those with many layers, known as deep neural networks or DNNs [39].

ANNs have been used with relative success in modeling and designing microwave passive circuits [40], checking the designed circuit's conformity [41], and designing simple circuits [42]. Using more layers, DNNs have also been successfully used to solve inverse problems (e.g. see [43]), and contextual problems such as language translation [44].

In the context of circuit sizing, DNNs present the potential to overcome the main limits of the methods described in the previous subsections. First, they do not have the explicit knowledge acquisition problem of knowledge-based approaches, as they learn autonomously from the example data. Next, the training data come from already sized circuits that are used as examples, and the optimized solution corresponds to a generic mapping of performance criteria to circuit sizing, instead of being idiosyncratic. Hence, once trained, they do not need to restart for each new set of performance criteria. Therefore, the time invested in sizing a set of circuits for training a DNN is largely repaid by reuse in the long run.

However, the number of already sized circuits to train a DNN for efficient prediction is a problem. Because the number of parameters (i.e. connection weights at the input of each neuron) increases as a power law with the number of neural layers, a huge number of examples is usually required for DNN training. Unfortunately, multiple reasons make this difficult, if not impossible, to achieve for analog circuit sizing. First, it takes time to synthesize each training example and there are no publicly available datasets; second, the technology for analog circuit design and implementation is not static, making it necessary to adapt the training sets to each new technological advance.

The lack of availability of large datasets of successfully sized microelectronic analog circuits and the search for a component sizing methodology that is generic in scope are the main motivations for the work presented here. Indeed, when looking at the literature, even in the few cases where researchers gathered tens of thousands of examples to train their DNNs, their models address low to mid-frequency analog circuits, operate within fixed design parameters or use data augmentation techniques with lower performance specifications to train the networks (e.g. [45]-[47]). An exception is the work in [48] whose two-model approach is somewhat like ours in that it tries to shrink the solution space before the final classification. But the work is mainly a proof of concept as presented and our approach is simpler. As will be presented next, our proposed method uses simple means to greatly reduce the required number of example data to train a neural network for circuit sizing, and it applies to arbitrary performances given a circuit topology.

III. METHODOLOGY

As argued above, DNNs are an attractive method for circuit sizing, but the relatively small size of the available training sets prevents their efficient training due to the large number of parameters to set. The proposed approach circumvents the problem by successively predicting the outputs one at a time: instead of a static DNN architecture to predict all the component sizes at once, the architecture is generated in steps, using a DNN made of a cascade of shallows neural networks (C-SNN). Two variants are presented: a fixed cascade with one SNN per size to predict (FC-SNN), and a dynamic cascade where more than one SNN contributes (DC-SNN). They are described next, along with the GA to optimize the hyperparameters of each SNN, and the validation method used in this work.

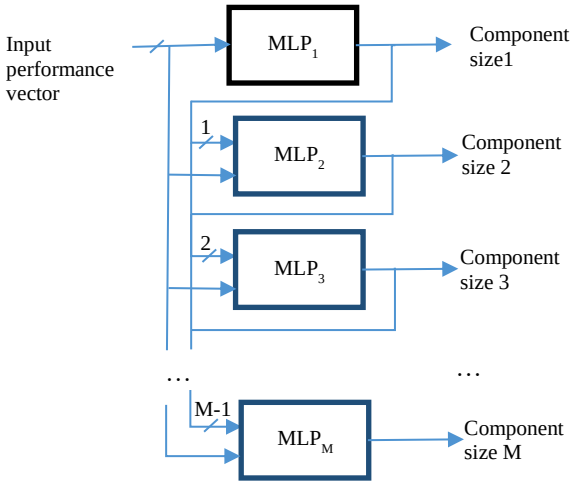

Figure 1. Block diagram of the C-SNN [49], where *MLP* means multilayer perceptron ANN and *M* is the number of component sizes to predict.

A. *Fixed cascade of shallow neural networks (FC-SNN)*

This is the base version of C-SNN, with each component size predicted without concern for the next sizes to predict or their couplings. Therefore, FC-SNN comprises as many SNNs as there are component sizes to predict as shown in Fig. 1.

Each SNN in the figure has one or two hidden layers, with each input layer neuron holding one input value, and each hidden layer neuron *j* producing an output given by:

$$y_j = f_\theta(s_j) = f_\theta\left(\sum_{i=1}^{N} \omega_{ji} x_i\right) \quad (1)$$

where s_j is the weighted sum of the *N* inputs from the previous layer, x_i is the output of the i^{th} neuron in that layer, ω_{ji} is a weight to be determined and f_θ is the neural output function. In this work, that function was either the hyperbolic tangent sigmoid:

$$f_{tanh}(s) = \frac{2}{1+e^{-2s}} - 1 \quad (2)$$

or the rectified linear unit (ReLU):

$$f_{ReLu}(s) = \begin{cases} s, & s > 0 \\ 0, & s \leq 0 \end{cases} \quad (3)$$

The type of output function and number of hidden neurons are set by the GA, and the same output function applies to all the hidden neurons in a given MLP.

In the output layer, the neural output is the weighted sum of its inputs, given by:

$$z = \sum_{j=1}^{N} \omega_{kj} y_j \quad (4)$$

where y_j the output of the j^{th} hidden layer neuron from the previous layer and ω_{kj} another weight to be determined by the learning process.

The neural weights are optimized using the error backpropagation with gradient descent algorithm, which minimizes the network's output error by propagating it back through the hidden layers and adjusting the different connection weights for minimal contribution to the error [50]. The gradient descent algorithm used the Adam optimizer [51] with regularization for weight setting, and the relevant neural hyperpaprameters set by the aforementioned GA. After tuning and subsequent training, each MLP in Fig. 1 adds its output to the specified performances to constrain the learning of the next MLP to find a new component size while compensating for potential coupling with the already predicted values.

The FC-SNN can be seen as a hybrid architecture that combines a GA and an error backpropagation with a gradient descent algorithm, working together to optimize the specification of each MLP stage. At the top level, the GA searches the space of MLP hyperparameters for the optimal values of the number of layers, the number of neurons per layer, the neural output functions, and the training algorithm's parameters. At the bottom level, the backpropagation algorithm with gradient descent searches the space of MLP weights to optimize the prediction of the selected target component size.

FC-SNN operates as follows: MLP_1 takes as inputs the performances of the desired circuit and the GA tunes it to predict a first component size. Then, the process repeats to optimize MLP_2 for the second component size to predict. MLP_2 takes both the performance criteria and the output of MLP_1 as inputs. In the next step, the performance criteria and the outputs of MLP_1 and MLP_2 form the input of MLP_3 in the FC-SNN sequence, and the process continues with each remaining MLP until all the component sizes have been correctly predicted.

One important issue is the component size prediction order. This is accomplished as follows: initially, several chromosome populations are randomly generated, one for each component size to predict. Then the GA starts with the first population to tune MLP_1 to predict each component size in turn. The GA is iterated until the 5% prediction accuracy threshold is reached and the best predicted component size is selected as the first one to predict. Then, the process repeats for the remaining component sizes using the second chromosome population and MLP_2, and the winner is selected as the second component size to predict. The same procedure is repeated for each of the remaining MLPs until no component size to predict is left, leading to an ordered prediction sequence. Subsection C.4 summarizes the algorithm using pseudo-code.

B. *Dynamic cascade of shallow neural networks (DC-SNN)*

Because of its sequential prediction, the FC-SNN method may lead to a deadlock when two component sizes are interdependent since each one must be known to determine the other. To overcome that problem, a new version of C-SNN is proposed, called the dynamic cascade of shallow neural networks (DC-SNN). It applies when one or more SNN in FC-SNN has a prediction error higher than the set threshold (5% component and performance tolerances in this work).

In DC-SNN, more than one SNN may contribute to sizing the same component, with the added SNNs accounting for potential component cross-couplings. The rationale behind this is that, while the SNN that predicts the first component size can only rely on the specified performance criteria for the target circuit as inputs, the same SNN moved *l* positions forward in the cascade can count on both the performance criteria and the *l*-1 size predictions obtained up to that point, hence potentially producing a lower prediction error.

The DC-SNN method starts by generating a cascade of *M* MLPs like FC-SNN, with MLP_i predicting the i^{th} component size d_i. Then, a new MLP, MLP_{i+M}, tries to learn d_i again using all the performance criteria and predicted values as inputs. If MLP_{i+M} predicts d_i better than MLP_i, it is added to the cascade as the predictor of d_i; otherwise, it is ignored. The DC-SNN approach is illustrated in Figure 2 for $M=3$.

C. *Genetic Algorithm for MLP Configuration*

As mentioned, a genetic algorithm is used to optimize the hyperparameters of each MLP in the cascade. Each GA starts by generating a population of chromosomes that encode the

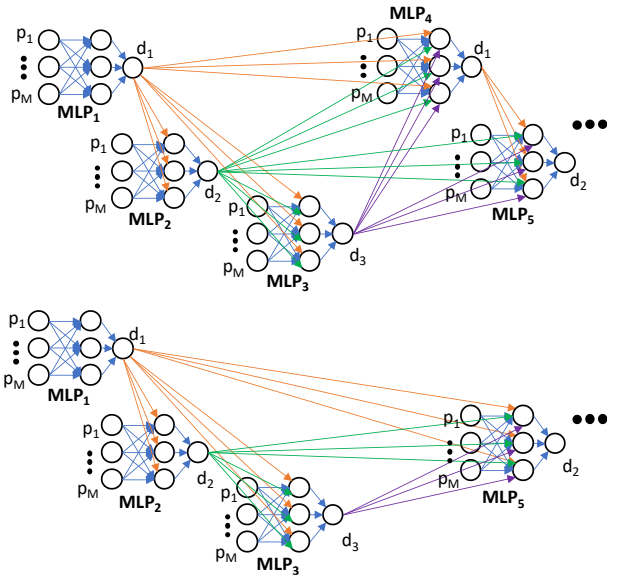

Fig. 2. Example of MLP selection in a DC-SNN model with 3 component sizes to predict, where *p* is a performance criterion and *d* a component size to predict. a) case where MLP_4 makes a better prediction of d_1 than MLP_1, leading to adding MLP_4 to the MLP sequence; b) case where MLP_4's prediction is worse, leading to not adding MLP_4 to the MLP sequence.

MLP hyperparameters as a binary vector where the successive bit fields represent different hyperparameters (see Table I). In this work, a population size of 64 randomly initialized chromosomes of 19-bit size is used. Then, the GA's three-step process of selection, reproduction and mutation is repeated until reaching a stopping criterion.

Table I. Description of chromosome bits for the genetic algorithm

Bit #	Role in MLP	Range of values
1 – 3	Nb. of hidden layer neurons	10, 15, …, 45
4	Nb. of hidden layers	1, 2
5	Hidden layer output function	Tanh, ReLU
6 – 8	Size of minibatch	1 - 8
9	L1 regularization	yes/no
10	L2 regularization	yes/no
11 – 13	β1 of Adam optimizer	0.9 - 0.99999
14 – 16	β2 of Adam optimizer	0.9 - 0.99999
17 – 19	ε of Adam optimizer	10^{-10} - 10^{-6}

1) *Selection Phase*

This phase selects the chromosome pairs for reproduction. The deterministic tournament method is used as it has been shown to be more efficient and less time-consuming than other selection methods such as ranking and roulette [52], and it has been suggested that its deterministic ranking has the potential to outperform probabilistic selection [53]. Moreover, deterministic tournament lends itself more easily to hardware implementation in a field-programmable gate array (FPGA), which is a future objective of this work.

In the deterministic tournament, the future parent chromosomes are selected through a series of pairwise matchups, using the mean-squared error between the target and predicted outputs of the associated MLP as a fitness function. For each matchup, the chromosome with the best fitness value proceeds to the next stage [54]. The chromosome population is arbitrarily ranked for fitness at the beginning.

2) *Reproduction Phase*

At the end of the selection phase, 32 of the 64 chromosomes have been deleted and the remaining 32 are re-ranked. Then, to replenish the population, 32 new chromosomes are generated from the remaining ones used as parents, with each pair breeding two children. Each child is created by using two randomly selected crossover points as is commonly done. The chromosome corresponding to the first child is identical to the chromosome of its first parent before the first crossover point and after the second crossover point and is identical to the chromosome of its second parent in between. The inverse is true for the second child of the same pair of parents.

3) *Mutation Phase*

The mutation phase inverts each chromosome bit of the population with probability $1/nb$, where *nb* is the size of the chromosome in bits. This probability was suggested in [53], and it was tested here with various other values in pilot experiments; none consistently provided better results.

4) *Final DC-SNN architecture*

The pseudo-algorithm of DC-SNN generation is as follows:

Algorithm 1. Genetic algorithm applied to the DC-SNN.

```
Import dataset;
Normalize dataset (min-max normalization);
Shuffle dataset order;
Split dataset into a training set, a validation set and a test set;
Generate N populations of M chromosomes, where N = number
of design sizes to predict and M=64;
while max(best validation_error for each size to predict)
                                                    > 0.05 do
    for i = 0, i < N do
        for j = 0, j < M do
            Fix the hyperparameters of new_MLP_ji for chromosome j
                                                    in population i;
            while validation_error improves do
                Load training and validation set using MLP sequence
                Shuffle training set order.
                Train new_MLP_ji;
                Validate new_MLP_ji;
                if validation_ error of new_MLP_ji < validation_ error
                                                    of old_MLP_ji do
                    append MLP_ji to MLP sequence
    Selection phase on each population;
    Reproduction phase on each population;
    Mutation phase on each population;
Test MLP sequence
```

IV. VALIDATION

The performances of FC-SNN and DC-SNN were tested with three different analog RF microcircuits: a low-noise amplifier (LNA), a voltage-controlled oscillator (VCO) and a mixer. For each circuit, 200 designs were completed with Cadence Cadence Virtuoso Spectre RF to serve as training examples, using randomly chosen, wide-ranging performance specifications. Moreover, to get a perspective on the obtained results, they were compared to those produced by a set of independent MLPs (I-MLP), each one only predicting one

component size from the performance specifications. This makes it possible to evaluate the impact of increasing the communication between MLPs on the sizing performance, starting with an architecture with no communication (I-MLP), to one with only successful MLPs communicating (FC-SNN), to one with all MLPs communicating (DC-SNN). The hyperparameters of the different models were all optimized by GA as described in Methodology.

1) *Low-Noise Amplifier*

The circuit topology of the LNA is shown in Fig. 3. It is the same as in [55] and consists of a cascode common-source stage with source degeneration and inductive load. The cascode configuration was selected for its design simplicity (easier matching, stability, etc.) and widespread use. All the inductances, including the matching networks', were assumed to be non-ideal with Q-factors of 10, which is consistent with current CMOS fabrication processes. To further simplify the design effort and reduce the number of parameters, only the L-shaped matching network shown in Fig. 3b is considered for 50 Ω source and load impedance matching. The source inductance (L_S) was also fixed at 0.578nH for all designs.

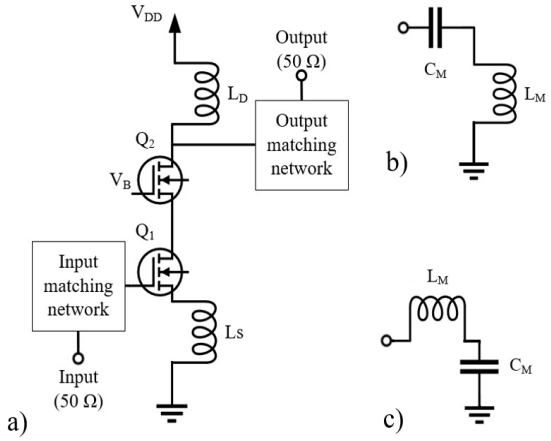

Fig. 3. LNA topology (a), input matching network (b) and output matching network (c) [55].

Table II. Considered ranges of performance parameters and design variables for the LNA

	Name	Min. value	Max. value
Performance	Bandwidth (MHz)	387.0	882.3
	1-dB compression Point (dB)	-21.04	-13.03
	Center Frequency (GHz)	2.07	4.76
	IIP3 (dB)	-11.58	2.64
	Noise Figure (dB)	1.386	3.33
	S21 (dB)	9.11	15.29
Design	Input capacitance (fF)	238.2	1528.2
	Input Inductance (nH)	1.811	10.610
	Output capacitance (fF)	260.0	2902.6
	Output Inductance (nH)	4.913	14.123
	Transistor Q_1 length (nm)	160	510
	Transistor Q_1 width (µm)	60	560
	Transistor Q_2 length (nm)	200	610
	Transistor Q_2 width (µm)	90	500
	Drain Inductance – L_D (nH)	1.707	8.671
	Bias voltage - V_B (mV)	300	600

Two hundred LNA microcircuits were designed using 180 nm CMOS technology. Table II shows the performance (P) and design (D) variables of the different designs, with the considered ranges of values.

2) *Voltage-Controlled Oscillator*

The second circuit topology is the symmetrical cross-coupled VCO illustrated in Fig. 4. It is a common VCO configuration that allows almost rail-to-rail output swing, with the cross-coupled PMOS-NMOS pair helping to reduce $1/f$ noise [56].

Two hundred VCO microcircuits were designed in CMOS 130 nm technology. Table III shows the performance and design variables of the different designs, and the considered ranges of values. The current source and inductance were fixed before each design and given as input to the sizing process, and V_{tune} was varied within each design to get the tuning range.

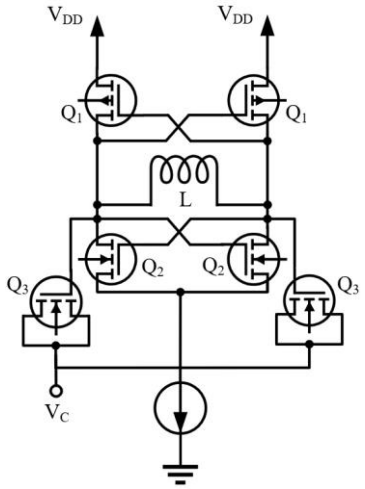

Fig. 4. VCO topology [56].

Table III. Considered ranges of performance parameters and design variables for the VCO

	Name	Min. value	Max. value
Performance	Oscillation frequency (GHz)	2.022	9.924
	Tuning range (MHz)	52.4490	723.40
	Phase noise (dB/Hz)	-102.7	-88.2
	Power consumption (mW)	1.386	3.33
Design	Transistor Q_1 length (µm)	0.5	4.1
	Transistor Q_1 width (µm)	2.5	55
	Transistor Q_2 length (µm)	0.5	4.4
	Transistor Q_2 width (µm)	3	55
	Transistor Q_3 length (µm)	0.8	4.2
	Transistor Q_3 width (µm)	15	98

3) *Mixer*

The circuit topology considered for the mixer is the double-balanced Gilbert cell presented in Fig. 5. It was selected because of its common use, since it provides good conversion gain and rejection at the input ports [56]. Moreover, the version used here includes a resonator that allows, through its parallel inductance L and capacitance C, to adjust the mixer for maximum response at the required frequency [57].

Here also, 200 mixer microcircuits were designed using CMOS 130 nm technology, with the lengths of all transistors fixed to 130 nm. Table IV shows the performance and design variables of the different designs, including the considered

ranges of values. Resistance *R* was fixed before the mixer sizing, and the size and gate voltage of transistor Q_3, which was part of a current mirror, were given as inputs in the sizing process.

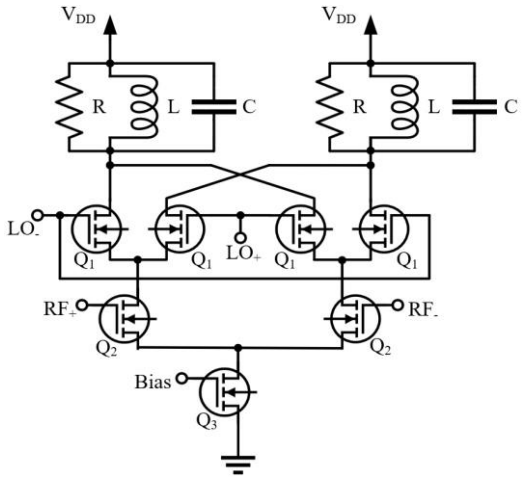

Fig. 5. Cross-coupled Gilbert cell mixer topology [57]

Table IV. Considered ranges of performance parameters and design variables for the mixer

	Name	Min. value	Max. value
Performance	RF frequency (GHz)	1.5	2.5
	IF frequency (MHz)	50	500
	Bandwidth (MHz)	48.68	409.1
	1 db compression point (dBm)	-18.69	-8.613
	IIP3 (dB)	-9.185	2.626
	Voltage gain (dB)	4.894	15.97
	SSB noise figure (dB)	4.464	5.637
	LO to IF isolation (dB)	-424.2	-169.3
	LO to RF isolation (dB)	-377.4	-278.4
	RF to IF isolation (dB	-235.8	-137
	Power consumption (mW)	8.560	47.712
Design	Capacitance (pF)	0.965	129.899
	Inductance (nH)	210	990
	Transistor Q1 width (µm)	200	430
	Transistor Q2 width (µm)	200	430
	Voltage at the gate of Q1 (V)	0.7	1.1
	Voltage at the gate of Q_2 (V)	0.65	0.9

4) Procedure

The prediction performances of the proposed FC-SNN and DC-SNN architectures, along with those of the I-MLP set of independent MLPs, were compared based on the designed LNA, VCO and mixer microcircuits. For each set of 200 designs previously described, 180 were used for training, 10 for validation and 10 for testing, with 2-fold cross-validation. The models were coded in Python using the Theano [58] and Lasagne [59] libraries. The simulations were run on a laptop computer with an Intel i7-7700HQ CPU clocked at 2.8 GHz, with 8 GB RAM and no dedicated graphics card. The genetic algorithms were set to run for a maximum of 10 generations.

The sizing performance of each trained model for the LNA, VCO and mixer was first assessed with three metrics: 1) the number of correctly predicted component sizes (i.e. with an error below 5 % of the normalized target sizes during the test phase), 2) The mean prediction error during the testing phase, which measures the generalization ability of the solutions within the performance ranges in tables II, III and IV, 3) the number of GA generations to reach the testing phase (i.e. get a size prediction error below 5 % for all components during the validation phase). However, as 5% component size tolerance does not necessarily lead to 5% performance tolerance in non-linear circuits, the predicted component sizes by DC-SNN were simulated in Cadence in additional experiments, and the obtained microcircuit performances were compared to the reference ones.

V. Results

Table V summarizes the obtained results, showing the DC-SNN success in predicting all the component sizes of the three different microcircuits with a mean error at test time smaller than 5 %. The worst prediction performance what that of I-MLP, showing the superiority of the FC-SNN and DC-SNN over predicting the component sizes independently from each other.

The learning time of DC-SNN was 4 minutes and 37 seconds for the LNA, 36 minutes and 21 seconds for the VCO and 47 minutes and 56 seconds for the mixer. Once trained, the tool took less than 5 seconds to size each of the ten test circuits for each of the three types. Hence, the response time was very fast to find the component sizes given a set of performance criteria.

To further confirm the robustness of the DC-SNN algorithm, it was run 20 times for the LNA circuit, leading to a mean prediction error of 3.39% with 0.6% standard deviation, and all 20 runs predicted the sizes of the components with less than a 5% error threshold.

Table V. Simulation results summary

Circuit	Neural network	GA generations for minimum prediction error	Predicted sizes within 5% error threshold	Mean error on test set
LNA	DC-SNN	1	10/10	0.0391
	FC-SNN	1	10/10	0.0402
	I-MLP	10	8/10	0.0690
VCO	DC-SNN	2	6/6	0.0429
	FC-SNN	10	4/6	0.0731
	I-MLP	10	3/6	0.0924
Mixer	DC-SNN	3	6/6	0.0487
	FC-SNN	10	4/6	0.0708
	I-MLP	10	3/6	0.0861

Table VI to Table VIII present examples of the mean absolute error (MAE) of the predicted component sizes and resulting performance of the LNA, VCO, and mixer microcircuits. As expected, the errors on the component sizes do not generate the same magnitude errors on the circuit performances, but the MAE remained under 5 % for both the component sizes and circuit performances of the three circuit topologies.

Table IX shows the average MAEs of both the predicted parameters and the simulated performances for the 10 predicted designs by the algorithm. These results show that the non-linear dependencies between the component sizes and the circuit performances are not important enough for the predicted sizes to degrade the desired performance beyond the 5% threshold. Then, using the predicted sizes, sweep analysis can be used for fine-tuning, or hindsight about the circuit behavior to change

specific component parameters for the desired performances.

A single deep MLP was also tested to predict all the component sizes of a circuit at once. This network was allowed between 20 and 50 hidden layers by the genetic algorithm and the vanishing gradient problem that often occurs in deep neural networks was attenuated by using residual connections [60]. However, its prediction accuracy was very poor and not worth reporting. This failure may be due to the small training set (200 circuit examples) in comparison to the large number of parameters to learn for such a network. Still, an earlier experiment with a shallow MLP was tested for the LNA with similarly poor results [54], hence emphasizing the importance of an architecture such as a C-SNN in the context of small training sets.

VI. Discussion and Conclusion

Our results show that the proposed deep neural network made of a cascade of separately trained shallow MLPs can successfully perform an initial sizing of the components of an analog RF circuit design, as it successfully did so for three different types of RF microcircuits within the set error margin for values and performances. More precisely, the FC-SNN variant predicted all the components sizes of an LNA within five percent error tolerance, and all components' sizes of a VCO and a mixer within eight percent error tolerance, and DC-SNN variant predicted the components sizes of all three circuit types within five percent error tolerance.

The developed architecture offers an effective way to

Table VI. Example of prediction errors and resulting performances for the LNA

Component size	Target value	Predicted value	AE (%)	Performance	Target value	Value from predicted sizes	AE (%)
L_D (nH)	4.706	4.486	4.67	Bandwidth (MHz)	510	480	5.88
Q_1 width (µm)	280	292.46	4.45	Noise Figure (dB)	2.308	2.287	0.91
Q_1 length (nm)	380	363.78	4.27	S21 (dB)	14.419	15.033	4.26
Q_2 width (µm)	400	413.57	3.39	IIP3 (dB)	-2.034	-2.147	5.55
Q_2 length (nm)	500	481.18	3.76	1-dB comp. point (dBm)	-17.284	-18.272	5.71
C_{IN} (pF)	1328.2	1367.34	2.95				
L_{IN} (nH)	4.734	4.821	1.84				
C_{OUT} (pF)	1430	1403.13	1.88				
L_{OUT} (nH)	9.278	9.728	4.85				
		MAE:	**3.56**			**MAE:**	**4.46**

Table VII. Example of prediction errors and resulting performances for the VCO

Component size	Target value	Predicted value	AE (%)	Performance	Target value	Value from predicted sizes	AE (%)
Q_2 length (µm)	1.5	1.575	5.00	Frequency of oscillation (GHz)	4.658	4.497	3.46
Q_2 width (µm)	26	27.014	3.90	Minimum frequency (GHz)	3.951	3.823	3.24
Q_1 length (µm)	2	1.9173	4.14	Maximum frequency (GHz)	5.467	5.302	3.02
Q_1 width (µm)	38	39.7405	4.58	Tuning range (GHz/GHz)	0.325	0.329	1.23
Q_3 length (µm)	2	2.08374	4.19	Phase noise (dBc/Hz @ 1MHz)	-108.6	-106.2	2.21
Q_3 width (µm)	80	83.644	4.56				
		MAE:	**4.39**			**MAE:**	**2.63**

Table VIII. Example of prediction errors and resulting performances for the Mixer

Component size	Target value	Predicted value	AE (%)	Performance	Target value	Value from predicted sizes
C (pF)	90	85.77	4.70	Bandwidth (MHz)	45.52	46.26
L (nH)	28.14	29.07	3.30	1-dB comp. point (dBm)	-15.91	-15.38
R (Ω)	400	411.5	2.88	IIP3 (dBm)	-4.989	-6.02
Q_1 width (µm)	400	388.58	2.86	Voltage gain (dB)	13.13	13.62
Q_2 width (µm)	430	450.76	4.83	SSB noise figure (dB)	4.411	4.403
Q_3 width (µm)	370	361.99	2.16	LO to IF isolation (dB)	-119.937	-122.56
V_{BIAS} (V)	0.58	0.605	4.31	LO to RF isolation (dB)	-256.5	-256.5
V_{RF} (V)	0.76	0.735	3.29	RF to IF isolation (dB)	-120.36	-124.77
V_{LO} (V)	1.04	1.015	2.40			
		MAE:	**3.41**			**MAE:**

Table IX. Test set MAE of the sizing predictions and resulting performances for the LNA, VCO and mixer circuits

Design Number	LNA		VCO		Mixer	
	MAE of parameters (%)	MAE of performances (%)	MAE of parameters (%)	MAE of performances (%)	MAE of parameters (%)	MAE of performances (%)
1	3.56	4.46	4.39	2.63	3.41	4.42
2	4.64	4.07	3.14	4.41	4.56	3.21
3	3.19	4.67	2.22	4.49	3.17	3.34
4	4.7	4.54	2.88	3.21	3.14	4.51
5	2.88	3.25	4.76	3.45	2.25	3.87
6	4.41	3.27	3.23	4.31	3.66	3.57
7	2.99	3.41	2.37	3.19	4.37	3.19
8	3.73	3.86	4.45	4.55	2.58	4.18
9	4.64	3.95	4.49	3.9	3.73	3.36
10	4.42	4.58	4.34	3.22	3.56	2.76
Average MAE (%)	**3.92**	**4.01**	**3.63**	**3.74**	**3.44**	**3.64**
Standard Deviation	**0.73**	**0.55**	**0.96**	**0.68**	**0.71**	**0.58**

circumscribe the neighborhoods of the sought component values in solution space, and the presented work is an efficient first step in a two-step solution to the sizing problem, where the second step fine-tunes the predicted values to account for layout and EM issues. As mentioned in Introduction, this second step is still a challenge as no automatic procedure exists for arbitrary circuit topologies and performances. Ongoing research to automate the whole process includes techniques based on surrogate models [30], Bayesian fusion [38], layout migration and reuse [61], layout self-organization [62], reinforcement learning [63], and training our deep neural network using a database of successfully fabricated circuits. However, the current cost of IC fabrication prevented us from creating such a database, given the number of end-to-end designs that must have been completed.

The proposed neural network-based methodology distinguishes itself by the relatively small training set required for training, thus circumventing the difficulty to gather the large number of microelectronic circuit designs normally required to train a deep neural network, a problem that does not exist in other domains that use deep learning such as image classification or language translation. Here, only two hundred exemplars were included in each dataset to learn the final MLP sequences. This was made possible by predicting the component sizes one by one and using MLPs with only one or two hidden layers, hence reducing the network complexity and quantity of parameters to learn at each step. That allowed a progressive process of constrained predictions to take place, with a relatively small training set used to solve the circuit sizing problem.

Another distinction of the proposed method from the related work is to go beyond finding idiosyncratic component values, as it creates a general mapping from the desired performance criteria to the component sizes without being specific to any set of performance values. Indeed, although the exemplars used for testing the model were not seen by the algorithm during the training or validation phases, there was no need to retrain the network for each one of them.

In this respect, the proposed approach is useful to generate a C-SNN that provides the initial sizing conditions to a more precise idiosyncratic optimization method. This would reduce the convergence time of those methods while preserving their accuracy.

The final component values obtained during the test phase were not fabricated and thus, the corresponding specifications were not tested post-fabrication. At the present stage, the goal of the proposed method is mainly to help designers decrease the time needed to reach the final values of their analog RF circuit components for IC implementation, by saving lengthy simulation time for the initial sizing before layout. Moreover, as different designers have different styles, the component sizes yielded by the algorithm reflect the design experience and bias of the designers of the microelectronic circuits used for training the algorithm. In the future, standardized databases might be useful to minimize this potential bias. The tool could also be trained with a more complete database including the extracted parasitic components and it could be constrained to specific inductances that were designed with EM simulations, however, such a training database is difficult to build as already discussed.

Finally, the obtained results show that the proposed method can be applied to many different types of circuits and technologies with appropriate training. Thus, it generalizes well. Here, it was successful in correctly predicting the component sizes of an LNA, a VCO, and a mixer, where the first was designed using CMOS 180 nm technology, and the other two using CMOS 130 nm technology, but different circuits, topologies, and CMOS technologies could have been used as well and for arbitrary performances, by using the appropriate completed designs as training examples.

REFERENCES

[1] J. Scheible, and J. Lienig, “Automation of analog IC layout: challenges and solutions”. *Int. Symp. Physical Design (ISPD '15)*. pp. 33-40, 2015.

[2] J. B. Keller, “Inverse Problems,” *The American Mathematical Monthly*, vol. 83, no. 2, pp. 107–118, 1976.

[3] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005.

[4] J. Hadamard, Lectures on Cauchy’s problem in linear partial differential equations. New Haven Yale University Press, 1923.B. J. Sheu, J. C. Lee, and A. H. Fung, “Flexible architecture approach to knowledge-based analogue IC design”, *IEE Proceedings G - Circuits, Devices and Systems*, vol. 137, no. 4, pp. 266–274, Aug. 1990.

[5] I. J. Bahl, *Lumped elements for RF and microwave circuits*. Boston: Artech House, 2003.

[6] "Cadence to Acquire Neolinear." https://www.cadence.com/en_US/home/company/newsroom/press-releases/pr-ir/2004/cadencetoacquireneolinear.html (accessed Aug. 15, 2022).

[7] "Solido Variation Designer," *Siemens Digital Industries Software*. https://eda.sw.siemens.com/en-US/ic/solido/variation-designer/ (accessed Aug. 15, 2022).

[8] "Home," *MunEDA - EDA Tools for process migration, sizing and verification of Custom IC*. https://www.muneda.com/ (accessed Aug. 15, 2022).

[9] CASS Talks 2022 - Helmut Graeb, Technical University of Munich, Germany - July 8, 2022, (2022). Accessed: Aug. 30, 2022. [Online Video] At: https://www.youtube.com/watch?v=MZN730AXCkY]

[10] G. Shi, “Toward automated reasoning for analog IC design by symbolic computation – a survey,” *Integration, the VLSI journal*, vol. 60, pp. 117–131, Jan. 2018.

[11] B. J. Sheu, J. C. Lee, and A. H. Fung, “Flexible architecture approach to knowledge-based analogue IC design,” *IEEE Proceedings G - Circuits, Devices and Systems*, vol. 137, no. 4, pp. 266–274, Aug. 1990.

[12] R. Harjani, “OASYS: a framework for analog circuit synthesis,” in *Proceedings., Second Annual IEEE ASIC Seminar and Exhibit*, 1989, pp. 1247-1266.

[13] A. Sasikumar, and R. Muthaiah, “Towards analog design automation using evolutionary algorithm: a review,” *Indian Journal of Science and Technology*, vol. 9, no. 39, pp. 1–12, Oct. 2016.

[14] M. Meissner and L. Hedrich, "FEATS: Framework for Explorative Analog Topology Synthesis," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 213-226, Feb. 2015, doi: 10.1109/TCAD.2014.2376987.

[15] Inga Abel and Helmut Graeb. 2022. FUBOCO: Structure Synthesis of Basic Op-Amps by FUnctional BlOck COmposition. ACM Trans. Des. Autom. Electron. Syst. 27, 6, Article 63 (November 2022), 27 pages.

[16] A. L. Kidd, Ed., Knowledge Acquisition for Expert Systems: A Practical Handbook. New York, NY, USA: Plenum Press, 1987.

[17] A. Aamodt, and E. Plaza, “Case-based reasoning: foundational issues, methodological variations, and system approaches," *Artificial Intelligence Communications*, vol. 7, no. 1, pp. 39-59, Mar. 1994.

[18] J. L. Kolodner, “An introduction to case-based reasoning,” *Artificial Intelligence Review*, vol. 6, no. 1, pp. 3–34, Mar. 1992.

[19] A. Al-Kashef, M. M. Zaky, M. I. Dessouky, and H. El-Ghitani, “A Case-Based Reasoning Approach for the Automatic Generation of VHDL-AMS Models,” in *BMAS 2008 - Proceedings of the 2008 IEEE International Behavioral Modeling and Simulation Workshop*, 2008, pp. 100–105.

[20] M. Fakhfakh, E. Tlelo, and P. Siarry, “Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design,” Springer, 2015.

[21] H. M. Venkataswamy and B. P. Harish, “AMSDAT: Integrated Analog and Mixed-Signal Design Optimization Framework for SoC

Applications," in 2021 IEEE 18th India Council International Conference (INDICON), Dec. 2021, pp. 1–6.

[22] E. Aarts, J. Korst, and W. Michiels, "Simulated Annealing," in *Search Methodologies*, Springer, Boston, MA, 2005, pp. 187–210.

[23] Y. Yang *et al*., "Smart-MSP: A Self-Adaptive Multiple Starting Point Optimization Approach for Analog Circuit Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 531–544, Mar. 2018.

[24] T. O. Weber, S. Chaparro, and W. A. M. V. Noije, "Synthesis of a narrow-band Low Noise Amplifier in a 180 nm CMOS technology using Simulated Annealing with crossover operator," in *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2013, pp. 1–5.

[25] A. A. Kalentyev, L. I. Babak, and D. V. Garays, "Genetic-algorithm-based sythesis of low-noise amplifiers with automatic selection of active elements and dc biases," in 2014 9th European Microwave Integrated Circuit Conference, 2014, pp. 520–523.

[26] G. I. Tombak et al., "Simulated annealing assisted NSGA-III-based multi-objective analog IC sizing tool," Integration, vol. 85, pp. 48–56, 2022.

[27] D. Whitley and A. M. Sutton, "Genetic Algorithms — A Survey of Models and Methods," in *Handbook of Natural Computing*, Springer, Berlin, Heidelberg, 2012, pp. 637–671.

[28] M. Barros, J. Guilherme, and N. Horta, "Analog Circuits Optimization Based on Evolutionary Computation Techniques," *Integr. VLSI J.*, vol. 43, no. 1, pp. 136–155, Jan. 2010.

[29] R. Zhou, P. Poechmueller, and Y. Wang, "An Analog Circuit Design and Optimization System with Rule-Guided Genetic Algorithm," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022.

[30] B. Liu, et al. "GASPAD: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 33.2 (2014): 169-182.

[31] J. R. Koza et al., "Synthesis of topology and sizing of analog electrical circuits by means of genetic programming," *Comp. Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 459-482, June 2000.

[32] T. Liao, and L. Zhang, "Parasitic-aware GP-based many-objective sizing methodology for analog and RF integrated circuits," presented at the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, Japan, 2017, pp. 475-480.

[33] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 109–128, Jul. 1997.

[34] J. D. Lohn and S. P. Colombano, "Automated analog circuit synthesis using a linear representation," in *Evolvable Systems: From Biology to Hardware*, 1998, pp. 125–133.S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, p. 671, May 1983.

[35] B. Liu, M. Zhongkun, G. A. E. Vandenbosch, G. Gielen , and P. Excell, "An efficient method for antenna design optimization based on evolutionary computation and machine learning," *IEEE Trans. Antennas & Propagation*, vol. 62, no. 1, pp. 7-18, Sept. 2014.

[36] B. Liu, Q. Zhang, and G. Gielen, "A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 2, pp. 180-192, April 2014.

[37] P. Chen, et al., "Bayesian optimization for broadband high-efficiency power amplifier designs." IEEE Transactions on Microwave Theory and Techniques 63.12 (2015): 4263-4272.

[38] F. Wang, W. Zhang, S. Sun, X. Li, and C. Gu, "Bayesian model fusion: large-scale performance modeling of analog and mixed-signal circuits by reusing early-stage data," in Proceedings of the 50th Annual Design Automation Conference, Austin Texas: ACM, May 2013, pp. 1–6. doi: 10.1145/2463209.2488812.

[39] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning," *Genet Program Evolvable Mach*, pp. 1–3, Oct. 2017.

[40] H. Kabir, L. Zhang, M. Yu, P. H. Aaen, J. Wood, and Q.-J. Zhan. "Smart modeling of microwave devices," *IEEE Microwave Magazine*, vol. 11, no. 3, pp. 105-118, May 2010.

[41] R. M. Hasani, D. Haerle, and R. Grosu, "Efficient modeling of complex analog integrated circuits using neural networks," presented at the *12th PRIME*, Lisbon, Portugal, 2016, pp. 1-4.

[42] A. Jafari, S. Sadri and M. Zekri, "Design optimization of analog integrated circuits by using artificial neural networks," presented at the *2nd SoCPaR*, Paris, France, 2010, pp. 385-388.

[43] J. Zhang, R. Su, Q. Fu, W. Ren, F. Heide, and Y. Nie, "A survey on computational spectral reconstruction methods from RGB to hyperspectral imaging," Scientific Reports, vol. 12, no. 1, 2022.

[44] Z. Ben Houidi and D. Rossi, "Neural language models for network configuration: Opportunities and reality check," Computer Communications, vol. 193, pp. 118–125, 2022.

[45] N. Kahraman and T. Yildirim, "Technology independent circuit sizing for fundamental analog circuits using artificial neural networks," *2008 Ph.D. Research in Microelectronics and Electronics*, Istanbul, 2008, pp. 1-4.

[46] N. Takai and M. Fukuda, "Prediction of element values of OPAmp for required specifications utilizing deep learning," *2017 International Symposium on Electronics and Smart Devices (ISESD)*, Yogyakarta, 2017, pp. 300-303.

[47] N. Lourenço *et al*., "On the Exploration of Promising Analog IC Designs via Artificial Neural Networks," *2018 15th International Conference[10 on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Prague, 2018, pp. 133-136.

[48] N. Lourenço *et al*., "Using Polynomial Regression and Artificial Neural Networks for Reusable Analog IC Sizing," *2019 16th Int. Conf. on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2019, pp. 13-16.

[49] E. Dumesnil, F. Nabki, and M. Boukadoum, "RF-LNA circuit synthesis using an array of artificial neural networks with constrained inputs," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 573–576.

[50] Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[51] D. P. Kingma, and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[52] L. J. Eshelman, "Genetic algorithms," in *Basic Algorithms and Operators* (1st ed.). Thomas Back, David B. Fogel, and Zbigniew Michalewicz (Eds.). IOP Publ. Ltd., Bristol, UK: 1999, pp. 64-80.

[53] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[54] E. Dumesnil, F. Nabki and M. Boukadoum, "RF-LNA circuit synthesis by genetic algorithm-specified artificial neural network," *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Marseille, 2014, pp. 758-761.

[55] M. Boukadoum, F. Nabki, and W. Ajib, "Towards neural network-based design of radiofrequency low-noise amplifiers," in *2012 IEEE International Symposium on Circuits and Systems*, 2012, pp. 2741–2744.

[56] B. Razavi, RF Microelectronics (2Nd Edition) (Prentice Hall Communications Engineering and Emerging Technologies Series), 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2011.

[57] X. Hu, "RF CMOS Tunable Gilbert Mixer with Wide Tuning Frequency and Controllable Bandwidth: Design Sythesis and Verification," *Browse all Theses and Dissertations*, Jan. 2017.

[58] Theano Development Team.: Theano: A Python framework for fast computation of mathematical expressions. (2016).

[59] S. Dieleman, J. Schlüter, C. Raffel, E. Olson. et al., "Lasagne: First release," DOI: http://dx.doi.org/10.5281/zenodo.27878, 2015.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[61] M. P. -H. Lin, Y. -W. Chang and C. -M. Hung, "Recent research development and new challenges in analog layout synthesis," 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macao, China, 2016, pp. 617-622, doi: 10.1109/ASPDAC.2016.7428080.

[62] D. Marolt, J. Scheible, G. Jerke and V. Marolt, "SWARM: A self-organization approach for layout automation in analog IC design", International Journal of Electronics and Electrical Engineering, vol. 4, no. 5, 2016.

[63] K. Settaluri, Z. Liu, R. Khurana, A. Mirhaj, R. Jain, and B. Nikolic, "Automated Design of Analog Circuits Using Reinforcement Learning," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 9, pp. 2794–2807, Sep. 2022, doi: 10.1109/TCAD.2021.3120547.