Design Automation of Analog and Mixed-Signal Circuits Using Neural Networks – A Tutorial Brief

Gustavo Liñán-Cembrano, Nuno Lourenço, *Member, IEEE*, Nuno Horta, *Senior Member, IEEE*, and José M. de la Rosa, *Fellow, IEEE*

***Abstract*—This tutorial brief shows how Artificial Neural Networks (ANNs) can be used for the optimization and automated design of analog and mixed-signal circuits. A survey of conventional and computational-intelligence design methods is given as a motivation towards using ANNs as optimization engines. A step-by-step procedure is described explaining the key aspects to consider in our approach, such as dataset preparation, ANNs modeling, training, and optimization of network hyperparameters. As an application, two case studies at different hierarchy levels are presented. The first one is the system-level sizing of Sigma-Delta Modulators ($\Sigma\Delta$Ms), where ANNs are combined with behavioral simulations to generate valid circuit-level design variables for a given set of specifications. The second example combines ANNs with electrical simulators to optimize the circuit-level design of operational transconductance amplifiers. The results validate the presented approach and show its benefits with respect to prior art on synthesis methods of analog and mixed-signal circuits and systems.**

***Index Terms*—Design Automation, Optimization, Neural Networks, Analog and Mixed-Signal Circuit Design.**

I. Introduction

PROMPTED by technology downscaling, modern electronic devices are predominantly constructed using digital circuits. These circuits are, *a priori*, more programmable, energy-efficient, and cost-effective and also offer greater robustness against device imperfections compared to analog circuits. The increasing prevalence of digital signal processing goes hand in hand with the advancement of synthesis methods and Electronic Design Automation (EDA) tools which assist designers in automating and optimizing circuit development all the way from initial specifications to the final silicon implementation. In contrast, the design of analog circuits still relies heavily on empirical rules and practical knowledge derived from prior experiences and expertise.

While analog design automation still lags far behind its digital counterpart, numerous EDA tools and design methodologies have been reported. The prevailing strategies often adopt an optimization-based design methodology, which involves employing a simulator as a performance evaluator, coupled with an optimization engine, to discern the optimal solution within the design space. In pursuit of this goal, a variety of optimization techniques, such as genetic algorithms and simulated annealing, among others, have been employed to guide behavioral simulators at the system level and electrical simulators at the circuit level [1].

Recent uses of Artificial Intelligence (AI) algorithms for automating analog circuit design have been reported [2]–[8]. In some works, the ANNs replace the simulator while in others they play the role of the optimization engine. In the latter, ANNs are trained to size systems according to specifications, automating the sizing process and generating optimal designs for previously uncharted specifications. This approach succeeded in designing analog circuits, e.g., operational amplifiers [2], and mixed-signal systems like Sigma-Delta Modulators ($\Sigma\Delta$Ms) [9], [10].

This brief builds upon the authors' previous work in [2], [10], and provides an overview on using ANN-driven methods for automated analog and mixed-signal circuit design. Two case studies demonstrate its application at different abstraction levels. At system level, ANNs combined with SIMSIDES [11] –a behavioral simulator– are employed for the high-level synthesis of $\Sigma\Delta$Ms. At circuit level, ANNs and SPICE-like simulators are used to automate transistor sizing and biasing of Operational Transconductance Amplifiers (OTAs).

The article is organized as follows. Section II overviews prior art on analog design methods and describes the proposed ANN-based synthesis methodology. The high-level synthesis of $\Sigma\Delta$Ms is presented in Section III. The use of ANNs to automate the design of basic analog circuits is covered in Section IV. Finally, conclusions are drawn in Section V.

II. Optimization-based Design Methodology using ANNs and Computational Intelligence Techniques

Figure 1 illustrates the widely employed *top-down/bottom-up* hierarchical design methodology for analog and mixed-signal circuits and systems. In this approach, a given system is partitioned into a number of *hierarchical or abstraction* levels that depend on the system complexity, but which typically comprises, at least, the four levels depicted in Fig. 1; system, circuit, device, and physical. This way, the design process is divided into several *sizing* steps, so that the system specifications are transmitted (or *mapped*) in a hierarchical way, from top (system) level to bottom (physical) level, and the performance is verified in the reverse path – from bottom to top [12], [13].

Manuscript received September 1, 2023; revised September 25, 2023.

This work was supported in part by Grant PID2019-103876RB-I00, funded by MCIN/AEI/10.13039/501100011033, by the European Union "ESF Investing in your future", and by "Junta de Andalucía" in Spain under grant P20-00599, by Fundação para a Ciência e a Tecnologia–Ministério da Ciência, Tecnologia e Ensino Superior (FCT/MCTES), in Portugal, through national funds and, when applicable, co-funded by European Union (EU) funds under the project UIDB/50008/2020.

Gustavo Liñán-Cembrano and José M. de la Rosa are with the Institute of Microelectronics of Seville, IMSE-CNM (CSIC/University of Seville), C/ Américo Vespuccio 28, 41092 Sevilla, Spain, e-mail: [linan,jrosa]@imse-cnm.csic.es. Nuno Lourenço and Nuno Horta are with the Instituto de Telecomunicações, Instituto Superior Técnico, Univ. de Lisboa, Portugal, e-mail: [nlourenco,nuno.horta]@lx.it.pt.

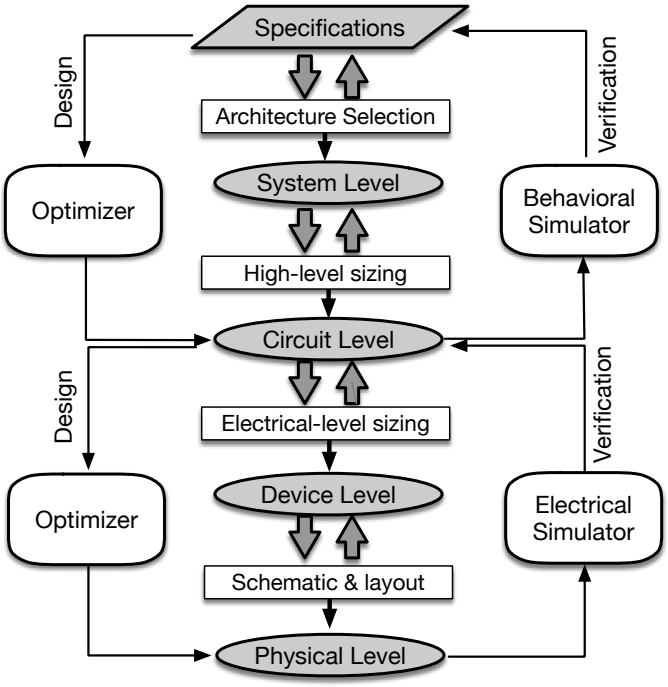

Fig. 1. Optimization-based top-down/bottom-up design process.

A. Conventional Optimization-based Design Method

At each abstraction level, a selection of the best topology (or architecture) is done. For instance, in the case of Analog-to-Digital Converters (ADCs), the system-level design involves selecting the best conversion technique, i.e. pipeline, SAR, $\Sigma\Delta$M, etc., as well as the more suitable topology within this family of converters for a given set of specifications. Something similar happens at circuit level, e.g. amplifiers, where the best suited topology, for instance, telescopic, folded-cascode, multi-stage, etc. needs to be chosen. Once a system or building-block topology has been defined, the next step consists of getting the design variables that satisfy the specifications at each abstraction level and propagate them through the hierarchy.

Two different sizing processes are shown in Fig. 1: system-level sizing and circuit-level sizing. The role of specifications and design variables changes as we move through the design flow. Thus, the design variables at a given level constitute the specifications for the level underneath. Let us illustrate this using one of the case studies in this tutorial, i.e. a $\Sigma\Delta$M. Here, the typical system-level specifications are the Effective Number of Bits (ENOB) and the signal bandwidth (BW). These specifications are mapped onto circuit-level requirements: i.e. amplifier DC gain, Gain-Bandwidth (GBW), Slew-Rate (SR), output swing, etc. These parameters, which play the role of design variables in the high-level sizing, turn into specifications at the electrical-design level. At this level, the design problem consists of obtaining the cell or circuit sizing and biasing i.e. the width, length and biasing of transistors and devices, which are the design variables at this step [14]. Both system- and circuit-level design processes are tasks that require multitude of *design/verification* iterations to meet the specifications [15]. To this purpose, different simulation approaches are needed to evaluate the performance of either the entire system or just a subsystem or building block. The latter can be analyzed using an electrical (SPICE-like) simulator offering high degree of accuracy. Conversely, the evaluation of the system-level performance requires an accurate but also CPU-time affordable solution. Here is where behavioral simulation comes into play by emulating the operation of the main building blocks using behavioral models that capture the most relevant circuit-level non-idealities, allowing a fast yet precise way to evaluate the performance of complex systems [11], [16]–[21]. One of these behavioral simulators is SIMSIDES [20], a time-domain simulator for $\Sigma\Delta$Ms developed in the MATLAB/SIMULINK environment, which will be used in this brief.

Typically, the sizing process is carried out by using an *optimization engine*, which guides the simulator through the exploration of the design space to find the optimum design solution. Different optimization algorithms have been proposed to this end, including genetic, simulated annealing, or multi-objective evolutionary Pareto fronts, to cite a few [11], [16], [19], [22], [23]. Regardless of the abstraction level, the optimization method involves defining a design-oriented *cost function* or a Figure of Merit (FOM) and minimizing it [24].

B. ANN-driven Optimization-based Design Method

Despite the advantages offered by the optimization-based approach, its primary obstacles lie in the large number of iteration cycles necessary for convergence and the significant CPU time needed to evaluate each potential solution.

ANNs can play a pivotal role in addressing these limitations by diminishing the necessity for resource-intensive simulations through: (1) selecting just the most promising candidate solutions for evaluation; (2) estimating some critical circuit performance targets, e.g. the performance in corners; or (3) replacing the circuit simulator entirely; alternatively, ANNs can also be employed to implement effective Reinforcement Learning (RL) agents that can size analog circuits or entirely replace the optimizer loop by inferring the values for the design variables directly from the specifications.

Figure 2 depicts the above mentioned approaches exemplified for a circuit-level design procedure. Figure 2(a) illustrates a general *Computational Intelligence* (CI) algorithm that uses the circuit simulator to evaluate the quality of the candidate designs. These automatic sizing methods aim to find a suitable set of device sizes by iterating over tentative guesses, by evaluating their performance using the circuit simulator, and incorporate the new data to select the next guesses. This method is widely used in academia and industry and produces valid designs, but it is very costly in terms of CPU time, and its reuse involves new optimization runs [1].

Several approaches leveraging ANNs which train models to predict the performance and replace the simulator [25]–[27] have been proposed, as illustrated by ANN-S model in Fig. 2(b). Although an ANN can be over $1000\times$ faster than the simulator, errors of more than 50% in some regions of the design space have been reported and many simulations are still needed to train the model. In a similar way, but

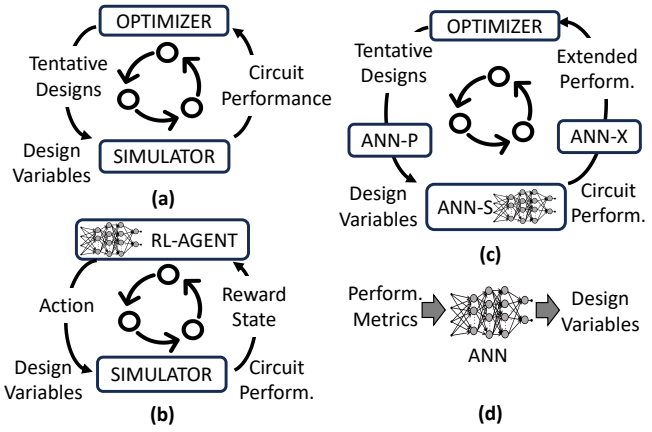

Fig. 2. Circuit design methods. (a) Simulation-based approach. (b) ANN-driven simulation-based approach. (c) RL approach. (d) Approach in this work.

without completely replacing the simulator, some approaches used an ANN to predict additional performances using at the input the design variables and a partial set of circuit measures obtained from the simulator, as illustrated by model ANN-X in Fig. 2(c). This method was used in [28] to bolster nominal simulations results predicting the performance figures over PVT corners. An alternative approach – denoted as ANN-P in Fig. 2(c) – keeps the simulator as performance evaluator but it trains some models to prune tentative solution guesses before being simulated [29], [30].

Figure 2(b) depicts an application of ANNs for circuit sizing under the umbrella of RL. Several works have shown that ANNs can be used to effectively learn good policies for analog circuit sizing, even considering robustness [6], [31]–[36]. While a trained agent can perform the sizing very accurately and effectively, defining an agent and reward functions suitable to be generally reusable over diverse circuit topologies and specifications is still an open topic. Figure 2(d) illustrates the approach of this tutorial, which consists of directly using the ANN in a supervised learning manner to produce the circuit sizing from the specifications. In this case, the ANN is trained to infer optimum designs as described below.

III. ANN-driven System-level Design: Application to ΣΔMs

As stated in Section II.A, the high-level design involves solving two types of problems. First, for a given vector of specifications, $\bar{\Gamma}$, a suitable architecture has to be selected from a family of alternatives $\{A_j\}$, where j represents available topologies. In the case of ΣΔMs, specifications typically involve ENOB, BW, Signal-to-Noise Ratio (SNR), Signal-to-Noise and Distortion Ratio (SNDR), Total Harmonic Distortion (THD), a FOM[1], etc., but they might also include constraints imposed by the system where the ΣΔM is to be embedded, by application-related scenarios, i.e. military, industrial, space, automotive, mass market, etc. Based on the selected topology, a vector of architecture-dependent design variables $\bar{\epsilon}(A_j)$ needs to be found so that specifications $\bar{\Gamma}$ are not only met but also optimized according to some predefined metrics.

[1]The Schreier's Figure of Merit – denoted as FOMS [37] and widely adopted by ΣΔM community – will be used in this work.

A. Problem Formulation and Definition of the Dataset

For the first problem, finding a suitable architecture from the information in the dataset, we propose to obtain inferences from a trained classifier [38], C, which maps system-performance metrics ($\bar{\Gamma}_i$) onto a categorical variable A_j (the selected architecture). For the second problem, a constrained optimization [39] problem which must yield optimal design variables for the given specifications and the chosen ΣΔM architecture, we propose to use Regression-type Neural Networks (RNNs), trained on comprehensive datasets, to infer the design variables.

This methodology can be formulated as follows: $A_j = C(\bar{\Gamma}_i)$; $\bar{\epsilon_k} = RNN(\bar{\Gamma}_i, A_j)$, where $C(\cdot)$ is a call to the classifier to obtain an architecture, and $RNN(\cdot)$ is a call to the RNN to infer the design variables. This way, every input in the dataset should be formatted as a triplet of the form $\{C_i, \bar{\Gamma}_i, \bar{\epsilon_i}\}$, where C_i is a categorical variable which defines the architecture of the modulator, $\bar{\Gamma}_i$ is a vector of ΣΔM performance metrics, and $\bar{\epsilon_i}$ is a vector which contains the design variables that produced such metrics in a behavioral simulation of this ΣΔM using SIMSIDES [11], [14].

B. Network Architecture and Model Optimization

Different classifier types were considered including Quadratic and Linear Discriminant Analysis (QDA, and LDA) [40], Support Vector Machines (SVM) [41] with linear, polynomial, and radial-basis function kernels, Gaussian and Multinomial Naive-Bayes (GNB, MNB) [42], Decission Trees (DT) [43], Random Forest (RF) [44], Gradient Boosting (GB) [45], and neural network-based classifiers [46].

A regression neural network [47] was used to infer the design variables for each modulator architecture. In order to optimize the computational performance of each RNN, Neural Architecture Search (NAS) techniques [48] were integrated in the design framework to automatically explore the *hyper-parameters* space and optimize the network architecture for each modulator using the Keras tuner [49]. The following hyper-parameters were considered for exploration: number of neurons (units) per layer; number of layers; use of dropout layers [50], activation function; and optimizer.

Due to the non-linear nature of ΣΔMs, it was crucial to determine how accurately the performance metrics $\bar{\Gamma}_k^{'}$ obtained for each inferred set of design variables $\bar{\epsilon}_k^{'}$ compared to the requested ones $\bar{\Gamma}_k$. To address this concern, a two-step approach was followed. First, we considered the network output as a coarse solution for the design variables, i.e. as a starting point. In a second step, we created 10 slightly varied versions of this point by randomly varying within $\pm 5\%$ each of the initially found design variables. Finally, the combination producing the best FOM [37] was selected.

C. ΣΔM Design Examples

Without loss of generality, we considered the following four ΣΔM architectures [10]: (I) a 2nd-order Switched-Capacitor (SC) ΣΔM; (II) a 3rd-order cascade 2-1 SC-ΣΔM; (III) a 4th-order cascade 2-1-1 SC-ΣΔM with 3-bit quantization; and (IV) a 2nd-order 3-level Continuous-Time (CT) ΣΔM based on Gm-C integrators. The specifications vector $\bar{\Gamma}$ was comprised of the required resolution (SNR), the bandwidth (expressed in terms of the OSR), and the power consumption $\bar{\Gamma}_k \equiv \{\text{SNR}_k, \text{OSR}_k, \text{Pow}_k\}$. The considered behavioral models included the main circuit nonidealities such as limited input/output swings, amplifier's input referred noise, finite DC gain, GBW, maximum output current and nonlinear transconductance. Effects like devices mismatch or switches KT/C noise were not included. In the case of SC-ΣΔMs in this case study, the design variables to be inferred were the finite DC gain, the finite transconductance, and the maximum output current of each amplifier. Simmilarly, for the Gm-C CT-ΣΔM we considered each amplifier's finite DC gain, GBW and third-order intermodulation product.

The dataset generation involved evaluating more than 200,000 random designs. The combined use of behavioral simulation (SIMSIDES), GPU acceleration and the MATLAB Parallel Computing Toolbox™ allowed us to obtain about 10,000 simulation results per hour of CPU time[2]. Needless to mention, we could not expect every randomly generated design to produce valid data for classifier and network training as they featured poor SNR. Therefore, all designs having SNR<50dB were removed from the dataset prior to the training. The final number of samples in the dataset amounted to nearly 120,000 $\{C_k, \bar{\Gamma}_k, \bar{\epsilon}_k\}$ elements that were minmax normalized to help training convergence. The dataset was partitioned into Training Set (TS) and Validation Set (VS) using the common $80\% - 20\%$ division. Moreover, 1000 dataset points (250 per modulator architecture) were reserved as Test Set and for comparison with other optimization approaches. After evaluating the performance of the different classifiers, we found that the Gradient Boosting classifier obtained the best classification score on the VS (93.7%), followed by the Random Forest (91.7%). Taking this into account, the GB classifier was chosen as the best suited for this case study. Regarding the neural network, the NAS algorithm was applied for each modulator (CPU times between $[1.9, 6]$h). Afterwards, each modulator' network template was fine-tuned for either 2000 epochs or until an early-stopping condition was met (CPU times between $[60, 397]$s). The MSE over the TS and the VS differed in less than 0.3% in all cases, showing no evidences of overfitting.

D. Verification and Comparison with Other Optimizers

In order to validate the method, we computed the relative deviation between the obtained specifications $\bar{\Gamma}'_k$, and the requested values $\bar{\Gamma}_k$ for the test set containing 1000 $\{\bar{\Gamma}_k, \bar{\epsilon}_k\}$ points. The RNN inference produced centered designs, i.e. designs that met specifications. Besides, the probability of producing designs surpassing the required specifications were of at least 68.5% in the worst case (2nd-ord Gm-C) and peaked to 78.5% for the cascade 2-1 SC. The solutions provided by the ANN were compared with established optimization algorithms for ΣΔMs [23] available in MATLAB®, namely Genetic, Gradient Descent, and Positive Basis Np1.

[2]All simulations were carried out in a PC with a i9-12900F@5GHz CPU, 64-GB RAM and NVIDIA® GeForce RTX™3060TI GPU.

TABLE I
COMPARISON WITH OTHER OPTIMIZATION ALGORITHMS

Architecture: 2nd-ord SC				
Algorithm	**CPU Time (min)**	**SNR (dB)**	***P* (mW)**	**FOM(dB)**
This work	0.05	86.01	10.16	149.0
Gradient	9	85.8	10.80	148.5
Genetic	18	88.1	18.41	148.5
Np1	5	86.6	20.00	146.6
Architecture: 2nd-ord GmC				
This work	0.05	75.19	0.07	182.6
Gradient	3	88.6	0.8	185.5
Genetic	129	90.2	0.8	187.0
Np1	10	88.0	0.4	187.9
Architecture: 3rd-ord Cascade 2-1 SC				
This work	0.05	115.8	2.7	181.5
Gradient	11	115.7	8.0	176.7
Genetic	145	116.2	4.1	180.1
Np1	12	115.9	8.0	176.9
Architecture: 4th-ord Cascade 2-1-1 SC				
This work	0.05	143.44	0.6	241.5
Gradient	4	144.32	1.8	237.9
Genetic	90	144.32	13.6	229.0
Np1	10	143.9	1.5	238.55

Table I[3] summarizes the results obtained by these optimizers when addressing the high-level sizing of the four ΣΔMs in our case study, considering an OSR $= 128$ and SNR of $\{88.1, 90.15, 116.2, 144.3\}$dB for the 2nd-ord SC, 2nd-ord Gm-C, Cascade 2-1 SC and the Cascade 2-1-1 architectures, respectively. The ANN-based methodology demonstrated a CPU time improvement of at least $60\times$ across the various modulators when compared to the considered optimizers, producing 20 specifications-design variables pairs per minute. Additionally, the presented methodology produced, in all cases, designs with a smaller power consumption and very similar SNR, yielding to better FOMs in most cases except in the 2nd-ord Gm-C ΣΔM.

IV. ANN-DRIVEN CIRCUIT-LEVEL DESIGN: APPLICATION TO OTAS

The system-level design in the previous section can be the starting point for a electrical, circuit-level design. Thus, for instance, the design variables obtained for the amplifiers can now play the role of specifications here, where the problem consists of obtaining transistor sizes and biasing for a set of requirements of a subcircuit or building block – an OTA in this case.

A. Preparing the Circuit-Design Dataset

The information used to train the ANN is again comprised of the set of data pairs denoted as $\{\bar{\Gamma}_i, \bar{\epsilon}_i\}$, but in this case $\bar{\Gamma}_i$ is a vector of the (OTA) circuit performance metrics (e.g., DC

[3]The high FOM in the 2-1-1 SC architecture is a consequence of the behavioral model not including switches KT/C noise nor device mismatches. However, since all the algorithms used the same behavioral model, the comparison is valid.

gain, GBW, noise figure, supply rejection ratio, etc.)[4], and, as before, $\bar{\epsilon}_i$, stands for the design variables, which at the circuit-level are typically, the transistors' multiplicity, biasing and sizing (channels length and width).

The ANNs are trained to predict the design variables $\bar{\epsilon}$, given a target vector of specifications, $\bar{\Psi}$, and only well-designed circuit sizings should be present in the dataset so that the ANN will predict near-optimal designs. Note that, while $\bar{\Gamma}$ can be used as the target specification, $\bar{\Psi}$, for a given point, it is not the most suited approach for analog IC sizing as specifications are usually defined as inequalities. Therefore, a suitable design is not only the one with the performance $\bar{\Gamma}$ exactly equal to $\bar{\Psi}$, but any solution with performance $\bar{\Gamma}'$ better than $\bar{\Psi}$. Moreover, if the models are trained only to map $\bar{\Gamma} \longrightarrow \bar{\epsilon}$, they cannot predict solutions for specifications outside the training set, even when the training set contains designs capable of meeting those specifications. However, since each sizing solution $\bar{\epsilon}_i$ corresponds to a circuit whose performance is $\bar{\Gamma}_i$, such a sizing is also a valid design for any performance target $\bar{\Psi}$ that is worse than $\bar{\Gamma}_i$. Therefore, the training data should be augmented with additional data pairs $\{\bar{\Psi}_i^k, \bar{\epsilon}_i\}$, where, each $\bar{\Psi}_i^k$ is the k^{th} randomized vector of performance values that are strictly worse (or covered by) than the real performance $\bar{\Gamma}_i$ of the solution point. As in the case of the $\Sigma\Delta$Ms, the dataset is normalized to improve training efficiency [2].

B. Defining and Using the Model

In this case, the network architecture is a dense (fully-connected) topology with 3-hidden layers. The data is split with a $80\% - 20\%$ ratio, and some additional design points, obtained independently, are used after training to verify the performance of the model outside the training data. To get predictions from the ANN we ask the model to make P predictions for $\bar{\Psi}_i^p$ randomly generated circuit performance targets, all of which are better than the desired specifications, e.g., given the specifications DC Gain$>$ 45dB, GBW$>$ 25MHz, $I_{DD} < 200\mu$A, then sets of inputs given to the ANN could be: $\{\{51\text{dB}, 33\text{MHz}, 190\mu\text{A}\}, \{75\text{dB}, 27\text{MHz}, 177\mu\text{A}\}, \ldots \{47\text{dB}, 25\text{MHz}, 200\text{A}\}\}$.

Although using the augmented datasets adds additional performance space coverage to the training data, it is also better to extend the coverage of the performance space during the sampling. Making the ANN to produce multiple predictions for each desired specification further attenuates the effect of the bias introduced during training and has a very small impact on the execution time due to the high parallel execution.

Once the ANN infers the design variables for all inputs, the corresponding designs can be simulated to obtain their true performance. Selecting a solution from this reduced set is a simple matter of establishing some single-valued metric or employing Pareto dominance to expose the trade-off between the relevant metrics.

[4] $\bar{\Gamma}_i$ is a vector which can be derived from the building-block design variables categorized as $\bar{\epsilon}_i$ in Section III.

TABLE II
PERFORMANCE OF SAMPLED DESIGNS

	Gain (dB)	**GBW (MHz)**	$\boldsymbol{I_{DD}}$**(µA)**	**PM**($^\circ$)	**FOM**[a]
Target 1	**50**	**60**	**300**	**65**	
Best FOM	51	63	325	65	1165
Target 2	**40**	**150**	**700**	**55**	
Best GBW	43	100	509	61	1182
Target 3	**50**	**30**	**150**	**65**	
Best I_{DD}	55	30	217	69	842
Best FOM	54	54	309	56	1050

[a] (MHz·pF/mA); The presented solutions all meet the constraints on overdrives and saturation with margins larger than 45mV.

C. OTA Design Example and Results

The OTA from [51] on a 130-nm CMOS node was considered as a case study. The dataset comprises $16,600$ design points obtained from several previously available optimization runs, and the circuit performances that were considered to train the ANN were DC Gain, I_{DD}, GBW, and Phase Margin (PM), which are extended with their second-order polynomial features.

The ANN has an input layer with 14 nodes (due to the polynomial features), 3 hidden layers with 120, 240, and 60 nodes each, and, an output layer with 12 nodes, corresponding to the widths and lengths of the 6 matched transistor pairs. It was trained on the original dataset, for 5000 epochs and a batch size of 512 in less than 15 minutes. Then, it was trained on a $40\times$ augmented dataset (nearly 700K samples) for 500 epochs, adding another 40 minutes to the training time [2].[5]

Table II shows the results of using the ANN to obtain the sizing for 3 different specifications targets. For each target, the ANN was asked to infer the transistor sizes for 100 performance vectors with random deviations of up to 15% from the target specifications. For each target, the ANN was sampled only once (with a batch size of 100). The election of the best solution was done by the FOM for target 1, GBW for target 2, and I_{DD} and FOM for target 3. The model succeeded in predicting the device sizes for the amplifier given their intended target performances, and circuits with FOMs larger than 1000 were obtained in all predictions. For target 3, two rows are listed to show how the ANN tried to meet the impossible specifications in different ways, showing that a properly trained ANN can generate a circuit sizing that is correct for specification trade-offs, including those not provided in the training data.

V. CONCLUSIONS

The use of ANNs for the automated design of analog and mixed-signal circuits and systems has been discussed in this tutorial brief. The presented methodology has been applied at the system-level for the synthesis of $\Sigma\Delta$Ms and at the circuit level for the optimization of OTAs. The obtained results are competitive with other optimization methods. Although still in its infancy, the use of ANNs to automate the design of analog and mixed-signal circuits may contribute to close the existing gap between digital and analog EDA tools.

[5] The model was implemented in Python with TensorFlow. Training was done on four Intel I7 cores@2.6GHz.

REFERENCES

[1] N. Lourenço, R. Martins, and N. Horta, *Automatic Analog IC Sizing and Optimization Constrained with PVT Corners and Layout Effects*. Springer, 2017.

[2] N. Lourenco *et al.*, "On the Exploration of Promising Analog IC Designs via Artificial Neural Networks," *Proc. of the 2018 Intl. Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2018.

[3] Y. Li *et al.*, "An Artificial Neural Network Assisted Optimization System for Analog Design Space Exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, pp. 2640–2643, October 2020.

[4] E. Afacan *et al.*, "Review: Machine learning techniques in analog/RF integrated circuit design, synthesis, layout and test," *Elsevier Integration, the VLSI Journal*, vol. 77, pp. 113–130, November 2021.

[5] M. Fayazi *et al.*, "Applications of Artificial Intelligence on the Modeling and Optimization for Analog and Mixed-Signal Circuits: A Review," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, pp. 2418–2431, June 2021.

[6] A. F. Budak, P. Bhansali, B. Liu, N. Sun, D. Z. Pan, and C. V. Kashyap, "Dnn-opt: An rl inspired optimization for analog circuit sizing using deep neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1219–1224, 2021.

[7] P. Jaraut *et al.*, "Augmented Convolutional Neural Network for Behavioral Modeling and Digital Predistortion of Concurrent Multiband Power Amplifiers," *IEEE Trans. on Microwave Theory and Techniques*, vol. 69, pp. 4142–4156, September 2021.

[8] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2018.

[9] J. M. de la Rosa, "AI-Assisted Sigma-Delta Converters – Application to Cognitive Radio," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, pp. 2557–2563, June 2022.

[10] P. Díaz-Lobo and J. M. de la Rosa, "High-Level Design of Sigma-Delta Modulators using Artificial Neural Networks," *Proc. of the IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, May 2023.

[11] J. Ruiz-Amaya *et al.*, "High-Level Synthesis of Switched-Capacitor, Switched-Current and Continuous-Time ΣΔ Modulators Using SIMULINK-based Time-Domain Behavioral Models," *IEEE Trans. on Circuits and Systems – I: Regular Papers*, pp. 1795–1810, Sep. 2005.

[12] V. F. Dias *et al.*, "Design Tools for Oversampling Data Converters: Needs and Solutions," *Microelectronics Journal*, vol. 23, pp. 641–650, 1992.

[13] G. Gielen and J. Franca, "CAD Tools for Data Converter Design: An Overview," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, pp. 77–89, February 1996.

[14] J. M. de la Rosa, *Sigma-Delta Converters: Practical Design Guide.* Wiley-IEEE Press, 2nd ed., 2018.

[15] F. V. Fernandez *et al.*, "Design Methodologies for Sigma-Delta Converters," *Chapter 15 in CMOS Telecom Data Converters (A. Rodríguez-Vázquez, F. Medeiro and E. Janssens, Editors)*, Kluwer Academic Publishers, 2003.

[16] K. Francken *et al.*, "A high-level simulation and synthesis environment for delta-sigma modulators," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, pp. 1049–1061, August 2003.

[17] R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters.* IEEE Press, 2005.

[18] S. Pavan, "Systematic Design Centering of Continuous Time Oversampling Converters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, pp. 158–162, March 2010.

[19] T. Bruckner *et al.*, "A GPU-Accelerated Web-based Synthesis Tool for CT Sigma-Delta Modulators," *IEEE Transactions on Circuits and Systems - I: Regular Papers*, vol. 61, pp. 1429–1441, May 2014.

[20] J. M. de la Rosa, "Design Automation of ΣΔ Converters: A Review of Modeling, Synthesis and Optimization Techniques," *Proc. of the IEEE Intl. Conf. on Electron Devices and Solid-State Circuits (EDSSC)*, October 2017.

[21] J. Wagner, M. Ortmanss, and J. M. de la Rosa, "Man or Machine – Design Automation of Delta-Sigma Modulators," *Proc. of the IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, pp. 4229–4232, May 2018.

[22] M. Velasco, R. Castro-Lopez, and J. M. de la Rosa, "High-Level Optimization of ΣΔ Modulators Using Multi-Objetive Evolutionary Algorithms," *Proc. of the IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, pp. 1494–1497, May 2016.

[23] B. Cortes-Delgadillo *et al.*, "Embedding MATLAB Optimisers in SIMSIDES for the High-Level Design of ΣΔ Modulators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, pp. 547–551, May 2018.

[24] F. Medeiro *et al.*, "A Vertically Integrated Tool for Automated Design of ΣΔ Modulators," *IEEE J. of Solid-State Circuits*, vol. 30, pp. 762–772, July 1995.

[25] H. Liu, A. Singhee, R. Rutenbar, and L. Carley, "Remembrance of circuits past: Macromodeling by data mining in large analog design spaces," in *Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324)*, pp. 437–442, 2002.

[26] G. Alpaydin, S. Balkir, and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 240–252, 2003.

[27] G. Wolfe and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 198–212, 2003.

[28] P. Vaz, A. Gusmão, N. Horta, N. Lourenço, and R. Martins, "Speeding-up complex rf ic sizing optimizations with a process, voltage and temperature corner performance estimator based on anns," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1570–1574, 2022.

[29] A. F. Budak, M. Gandara, W. Shi, D. Z. Pan, N. Sun, and B. Liu, "An efficient analog circuit sizing method based on machine learning assisted global optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 5, pp. 1209–1221, 2022.

[30] J. Domingues, A. Gusmão, N. Horta, N. Lourenço, and R. Martins, "Accelerating voltage-controlled oscillator sizing optimizations with ann-based convergence classifiers and frequency guess predictors," in *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 1–4, 2022.

[31] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "Autockt: Deep reinforcement learning of analog circuit designs," in *2020 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 490–495, 2020.

[32] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[33] Z. Zhao and L. Zhang, "Deep reinforcement learning for analog circuit sizing," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2020.

[34] K.-E. Yang, C.-Y. Tsai, H.-H. Shen, C.-F. Chiang, F.-M. Tsai, C.-A. Wang, Y. Ting, C.-S. Yeh, and C.-T. Lai, "Trust-region method with deep reinforcement learning in analog design space exploration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1225–1230, 2021.

[35] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, S. Sapatnekar, R. Harjani, and J. Hu, "A circuit attention network-based actor-critic learning approach to robust analog transistor sizing," in *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, 2021.

[36] W. Shi, H. Wang, J. Gu, M. Liu, D. Z. Pan, S. Han, and N. Sun, "Robustanalog: Fast variation-aware analog circuit design via multi-task rl," in *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)*, pp. 35–41, 2022.

[37] S. Pavan, R. Schreier, and G. C. Temes, *Understanding Delta-Sigma Data Converters.* Wiley-IEEE Press, 2nd ed., 2017.

[38] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[39] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[40] B. Ghojogh and M. Crowley, "Linear and quadratic discriminant analysis: Tutorial." https://arxiv.org/abs/1906.02590, 2019.

[41] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[42] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[43] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[44] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[45] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[46] J. M. Zurada, *Introduction to Artificial Neural Systems*. West Publishing Company, 1992.
[47] K. P. Murphy, *Machine learning: a probabilistic perspective*. The MIT Press, 2012.
[48] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," Google Research, 2017.
[49] Keras, *Keras API Reference*. [Online]. Available: https://keras.io/api/., 2021.
[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
[51] R. Povoa *et al.*, "Single-stage amplifier biased by voltage combiners with gain and energy-efficiency enhancement," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 3, pp. 266–270, 2018.