

SpringerBriefs in Applied Sciences and Technology
Computational Intelligence

João L. C. P. Domingues · Pedro J. C. D. C. Vaz ·
António P. L. Gusmão · Nuno C. G. Horta ·
Nuno C. C. Lourenço · Ricardo M. F. Martins

Speeding-Up Radio-Frequency Integrated Circuit Sizing with Neural Networks

SpringerBriefs in Applied Sciences and Technology

Computational Intelligence

Series Editor

Janusz Kacprzyk, Systems Research Institute, Polish Academy of Sciences,
Warsaw, Poland

SpringerBriefs in Computational Intelligence are a series of slim high-quality publications encompassing the entire spectrum of Computational Intelligence. Featuring compact volumes of 50 to 125 pages (approximately 20,000-45,000 words), Briefs are shorter than a conventional book but longer than a journal article. Thus Briefs serve as timely, concise tools for students, researchers, and professionals.

João L. C. P. Domingues · Pedro J. C. D. C. Vaz ·
António P. L. Gusmão · Nuno C. G. Horta ·
Nuno C. C. Lourenço · Ricardo M. F. Martins

Speeding-Up Radio-Frequency Integrated Circuit Sizing with Neural Networks

João L. C. P. Domingues
Instituto Superior Técnico
Instituto de Telecomunicações
Lisbon, Portugal

António P. L. Gusmão
Instituto Superior Técnico
Instituto de Telecomunicações
Lisbon, Portugal

Nuno C. C. Lourenço 
Instituto Superior Técnico
Instituto de Telecomunicações
Lisbon, Portugal

Pedro J. C. D. C. Vaz
Instituto Superior Técnico
Instituto de Telecomunicações
Lisbon, Portugal

Nuno C. G. Horta 
Instituto Superior Técnico
Instituto de Telecomunicações
Lisbon, Portugal

Ricardo M. F. Martins
Instituto Superior Técnico
Instituto de Telecomunicações
Lisbon, Portugal

ISSN 2191-530X ISSN 2191-5318 (electronic)
SpringerBriefs in Applied Sciences and Technology
ISSN 2625-3704 ISSN 2625-3712 (electronic)
SpringerBriefs in Computational Intelligence
ISBN 978-3-031-25098-9 ISBN 978-3-031-25099-6 (eBook)
<https://doi.org/10.1007/978-3-031-25099-6>

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

In the present day, the integrated circuit (IC) industry has, now more than ever, a massive demand for electronic devices not only for the consumer electronics markets but also in other industries such as medical, automotive, or security. Despite Moore's Law not really being observed anymore, the evolution of the IC industry is still clearly observable every year, with designers building increasingly more complex, power-efficient, and integrated systems. These systems often combine analog and digital sections, where most components are integrated into a single chip originating mixed-signal systems-on-a-chip. While these ICs are implemented mainly using digital circuitry, analog operations are still fundamental and irreplaceable. Additionally, technologies such as the Internet of things or fifth-generation broadband join millions of devices and sensors. All these applications continuously gather an increasing amount of data, posing unprecedented challenges to each element of the networks. Due to this, today's market is highly conditioned by the strong demand for high communication rates, large bandwidths, and ultra-low power consumptions, in which radio-frequency (RF) ICs play a critical role. However, analog design is unlike digital design, where an automated flow is established for most design stages. The absence of effective and established computer-aided design tools for electronic design automation of analog and radio-frequency IC blocks poses a significant contribution to their bulky development cycles, leading to long, iterative, and error-prone designer intervention over their entire design flow.

In the past years, automatic simulation-based sizing approaches became essential in designing analog and radio-frequency IC blocks for modern applications to ensure their robustness. However, optimizations considering process, voltage, and temperature (PVT) corners or layout still pose unprecedented challenges in applying these tools due to the high simulation times and different simulator convergence issues. Therefore, the work presented in this book addresses the automatic sizing of analog ICs assisted by deep learning and artificial neural networks on two fronts. First, it proposes two deep learning models to assist the PVT-inclusive simulation-based sizing process of radio-frequency ICs, specifically, voltage-controlled oscillators (VCOs). Given specific devices' dimensions, the first model classifies the likelihood of the circuit to convergence for nominal and PVT corner cases, bypassing

solutions that will hardly produce valuable information for the optimization process. The model also predicts the VCOs' oscillating frequencies for the mentioned conditions. The methodology is tested on a state-of-the-art VCO, reducing 19% of the workload of the circuit simulator, ultimately saving almost 5 days of computational effort and improving the optimization result. Secondly, a PVT regressor is proposed, which inputs the circuit's sizing and the nominal performances to estimate the PVT corner performances via multiple parallel artificial neural networks. Two control phases prevent the optimization process from being misled by inaccurate performance estimates. The proposed controlled PVT estimator is tested on two state-of-the-art VCOs, reducing the workload of the circuit simulator up to 79% while achieving a speed-up factor of $2.92 \times$, ultimately saving more than 16 days of computational effort. Both methodologies can be used simultaneously, and ultimately, they offer a unique opportunity to reuse valuable legacy data, often discarded in optimization environments.

Finally, the authors would like to express gratitude for the financial support that made this work possible. The work developed in this book was supported by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020 (including internal research project LAY(RF)²/X-0002-LX-20) and Research Grant FCT-SFRH/BD/07123/2021.

This book is organized into four chapters.

Chapter 1 presents an introduction to the analog IC design area and discusses how the advances in machine learning can pave the way for new EDA tools.

Chapter 2 presents a study of the available tools for analog design automation. Starting with an overview of existing works where machine learning techniques are applied to analog IC sizing.

Chapter 3 presents two artificial neural network models for analog IC design to be incorporated within simulation-based sizing loops. The first model classifies the convergence of the circuit for nominal and PVT corners, bypassing solutions that will hardly produce valuable information for the evolutionary kernel, and the second predicts the pre-defined simulator values for the previous conditions.

Chapter 4 presents a controlled PVT regressor based on an artificial neural network, also intended to be incorporated within simulation-based synthesis. This regressor estimates the complete set of PVT corner performances via multiple parallel networks.

Lisbon, Portugal

João L. C. P. Domingues
 Pedro J. C. D. C. Vaz
 António P. L. Gusmão
 Nuno C. G. Horta
 Nuno C. C. Lourenço
 Ricardo M. F. Martins

Contents

1	Introduction	1
1.1	Analog/RF Integrated Circuit Design Automation	1
1.2	Analog IC Design Flow	3
1.3	Machine Learning and Analog IC Sizing	4
1.4	Conclusion	6
	References	6
2	Background and Related Work	9
2.1	Knowledge-Based Sizing	9
2.2	Optimization-Based Sizing	9
2.2.1	Equation-Based Evaluation	9
2.2.2	Simulation-Based Evaluation	10
2.3	Machine Learning in Simulation-Based Evaluation	12
2.3.1	Types of Supervision	13
2.3.2	Simulation-Based Sizing Enhanced with SVMs	14
2.3.3	Simulation-Based Sizing Enhanced with ANNs	15
2.4	Other ML/DL Efforts on Analog/RF Sizing	18
2.4.1	Predicting Sizing from Performances	18
2.4.2	Reinforcement Learning	19
2.5	Case Study	20
2.5.1	Dual-Mode Class C/D VCO	20
2.5.2	Dataset Generation	22
2.6	Conclusion	24
	References	24
3	Convergence Classifier and Frequency Guess Predictor Based on ANNs	29
3.1	Contributions	29
3.2	Classifier and Regressor Based on Deep ANNs	30
3.2.1	Underlying Architectures	30
3.3	Training the Model in Isolation (Results Pre-integration)	32
3.3.1	Dataset Processing	32

3.3.2	Feature Engineering	35
3.3.3	Convergence Classifier and Its Hyperparameters	35
3.3.4	Regressor and Its Hyperparameters	39
3.3.5	Final Model Details	42
3.3.6	Discussion	44
3.4	In-the-Loop Integration	45
3.4.1	Class C/D VCO for 3.5-to-4.8 GHz @ 50% Threshold	48
3.4.2	Class C/D VCO for 3.5-to-4.8 GHz @ 75% Threshold	53
3.4.3	Class C/D VCO for 3.5-to-4.8 GHz @ 90% and 100% Thresholds	55
3.4.4	Analysis of the Points Fed to the Simulator	57
3.4.5	Plug-and-Play Class C/D VCO 2.3 GHz-to-2.5 GHz	59
3.4.6	Plug-and-Train Ultralow-Power Class B/C VCO	61
3.5	Conclusions and Future Research Directions	61
3.5.1	Conclusions	62
3.5.2	Future Work	63
	References	64
4	Process, Voltage and Temperature Corner Performance Estimator Using ANNs	67
4.1	Contributions	67
4.2	Controlled PVT Regressor Based on Deep ANNs	68
4.3	Training the Model in Isolation (Results Pre-integration)	69
4.3.1	Dataset Processing	69
4.3.2	Feature Engineering	72
4.3.3	Tuning Hyper-Parameters	73
4.3.4	Final Model Details	78
4.3.5	Test Results	80
4.4	In-the-Loop Integration	81
4.4.1	Class C/D VCO with PVT Estimator Working at 100%	86
4.4.2	PVT Estimator with Error Controller	89
4.4.3	Results with Controlled PVT Estimator	93
4.5	Conclusions and Future Research Directions	106
4.5.1	Conclusions	106
4.5.2	Future Work	107
	References	108

Chapter 1

Introduction



1.1 Analog/RF Integrated Circuit Design Automation

In the present day, the IC industry has, now more than ever, a huge demand for electronic devices not only in the consumer electronics markets but also in other industries such as medical, automotive, or security. Despite Moore's law not really being observed anymore, the evolution of the IC industry is still clearly observable every year, with designers building increasingly more complex, power-efficient and integrated systems. These systems often combine analog and digital sections, where most components are integrated into a single chip originating mixed-signal systems-on-a-chip (SoCs). While most functionalities are implemented using digital or digital signal processing circuitry, analog circuits are the bridge between digital circuitry and physical devices with a steady increase in connectivity needs. Even though analog circuits only occupy a small fraction of the SoCs, their design effort is disproportionately large, as illustrated by Fig. 1.1 [1]. According to [2], the global IC market was worth \$412.3 billion in 2019 and is expected to grow to \$502.94 billion by 2023, with analog components being present in more than 50% of the total IC shipments yearly. Besides, the strict time-to-market constraints and development costs make electronic systems' design challenging, being, therefore, fundamental to accelerate their development process as much as possible.

Plenty of EDA tools and design methodologies have been made available to cope with new capabilities offered by the integration technologies. However, there is still a considerable discrepancy between the analog and digital IC design tools. The gap between the number of existing EDA tools for digital and analog circuits is usually explained by the fact that the digital market is much larger, absorbing the available resources. It is also easier to express a digital system, which can be represented naturally in terms of Boolean expressions, whereas, on the analog side, their design is less systematic, more knowledge-based, and more heuristic [3]. Even though analog circuits only occupy a small fraction of these SoCs, they are responsible for most design errors and expensive redesigns/reruns. Therefore, economic pressure

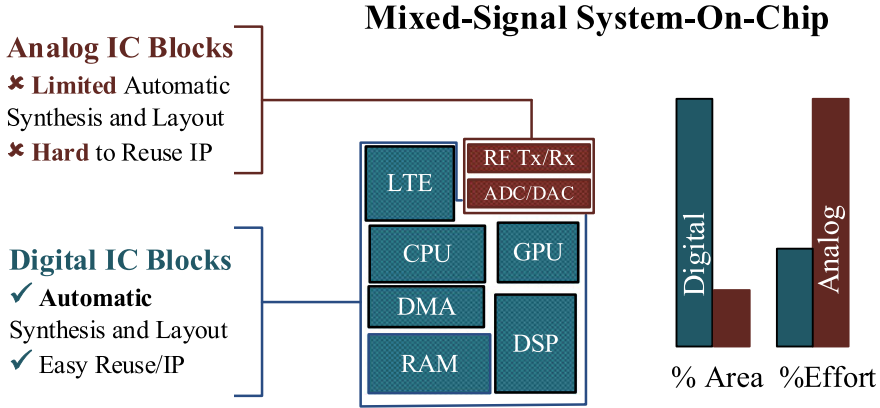


Fig. 1.1 Contrast between analog and digital blocks' design effort [1]

has motivated the quest for better methodologies to accelerate analog design. The automation level for analog IC has improved in the last few years, a field of profound academic and industrial research activity producing significant advances. However, it is still far from the push-button stage, leading designers to keep exploring the solution space almost manually, as there are no standard plug-and-play EDA tools and methodologies to automate the analog IC design flow.

While most of this is true for analog base-band, on top of that, with predictions that more than half of new businesses will run on the internet-of-things (IoT) and advanced telecommunication broadbands, such as the 5th generation (5G), there will be an immense demand for devices and sensors, opening doors to advances in many areas. This has already led to an increase in the amount of data that is being continuously generated, resulting in new challenges within every part of the network. Consequently, there is high pressure in today's market for larger communication rates, extensive bandwidths, and ultralow-power consumptions. This is where RF ICs come in hand, playing a crucial role. However, RF IC design in deep nanometric integration technologies for both IoT and 5G is extraordinarily difficult, due to their high complexity and demanding performances. Some of the design difficulties lie in the wide range of frequencies and dynamic ranges involved, but also on:

- Their dependence on non-reliable models of passive devices;
- At gigahertz frequencies, there is a huge impact of layout parasitics;
- Their integration in deep nanometer technologies causes variability issues and non-idealities which have never been experienced in older technology nodes.

Avoiding costly redesign cycles and reducing post-fabrication tuning and compensation work on first-pass fabrication success became primary RF IC design objectives. Established computer-aided design (CAD) companies provide environments that allow circuit designers to carry this flow manually. Despite this, the classical trial

and error method is no longer viable due to the high number of complex interactions leading to sub-optimal RF designs.

1.2 Analog IC Design Flow

Regarding the specific design flow of analog ICs, each designer/company may have its own. However, in [3], Gielen and Rutenbar standardized the steps that most designers take when designing an analog or mixed-signal IC, introducing the widely acknowledged design flow shown in Fig. 1.2. This flow consists of a series of top-down design steps repeated from the system to the device level and bottom-up layout generation and verification. Using a hierarchical top-down design methodology allows for performing system architectural exploration, achieving a better overall system optimization at a higher level of abstraction before starting more specific implementations at the circuit or device levels. With this, one can find problems earlier in the design flow, increasing the chances of first-time success with fewer time-consuming redesign iterations.

In this design flow, the number of hierarchy levels may vary according to the complexity of the system being designed, and although there are no overall accepted representations for the architectural design, the steps between two hierarchical levels are:

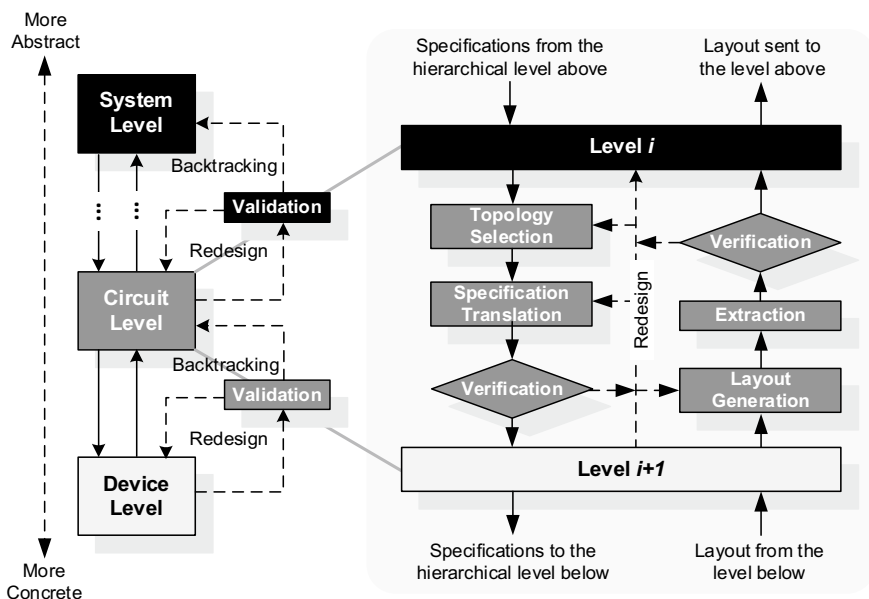


Fig. 1.2 Hierarchical levels and design tasks of analog design flow [4]

- Top-down electrical synthesis path, which includes topology selection, specification translation (or circuit sizing at the lowest level), and design verification;
- Bottom-up physical synthesis path, that includes layout generation and detailed design verification (after layout extraction).

In topology selection, the most appropriate circuit topology is determined to meet a certain number of specifications at the current hierarchy level. This topology can be chosen from a set of existing topologies or synthesized.

Specification translation is the step where the designer maps the high-level block specifications and, given a certain topology, maps them into individual specifications for each sub-blocks. Due to these sub-blocks being single devices at the lowest level, this task is narrowed down to circuit sizing. Before proceeding down in the hierarchy, specification translation is verified using simulation. At higher levels of the design flow, there is no device-level sizing available, which results in behavioral simulations. However, device sizing is available at lower levels (circuit and device level), and therefore, electrical simulations are used. Each block's specifications are passed to the following hierarchy level, and the whole process is repeated until the top-down electrical synthesis flow is completed.

To aid designers to overcome the many difficulties encountered in manual sizing of analog/RF IC blocks, several optimization-based sizing approaches emerged. These EDA tools use several algorithms that explore the design space effectively rather than iterating over designer-defined analytical equations. They can be used along with performance models that can capture several circuit characteristics of RF circuits. However, despite its increased computational effort, utilizing foundry-provided device models and a circuit simulator as an evaluation engine resulted in the most accurate and generally adopted approach. Most commercially available solutions that use the simulation-based architecture, e.g., Cadence's Virtuoso GXL [5] or MunEDA's DNO/GNO [6], still take a restrictive single-objective approach being used to semi-automate the manual sizing design process. Consequently, simulation-based techniques are a continuous research subject of the community to face the most recent design challenges.

After the top-down flow is completed at a certain level, the sizing obtained must be verified by generating the corresponding layout and testing its performance. If these prove to be satisfactory results, the design flow is finished. If not, a redesign is needed, repeating the previous steps or the complete flow.

1.3 Machine Learning and Analog IC Sizing

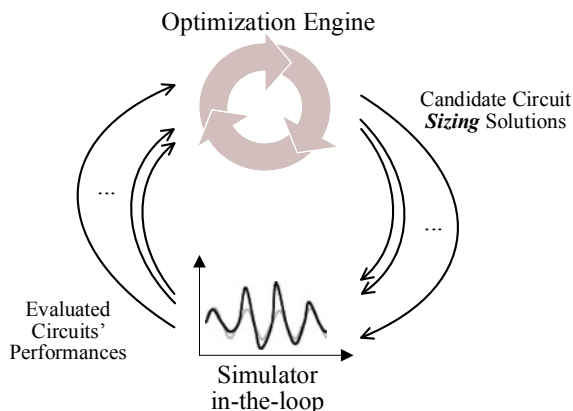
ML is how a computer improves its capabilities by analyzing past experiences. This area of artificial intelligence (AI) has been chosen to solve problems in computer vision, speech recognition, and natural language processing, among others because it can be easier to train a system showing examples of what the output should be given a particular input than to anticipate all possible responses for all inputs. One of the most

used models nowadays is artificial neural networks (ANNs), and those are the models considered in this book. This technique has been cyclically picked-up and abandoned over the years, but a new trend emerged recently, called deep learning (DL), where much more complex networks are employed, yielding fascinating results in image processing, for example. They are also playing a role in the design automation of analog and RF integrated circuits ICs [7], being used on several fronts, e.g., modeling [8], mapping from devices' sizes to circuits' performances [9–11], mapping from specifications to the sizing [12], layout generation [13–17] or even fault testing [18].

Analog IC performance evaluation is well established in the design flow. Furthermore, as stated in the previous section, this prominence of circuit analysis tools and methods led to simulation-based optimization as the most common method in both industrial [5, 6] and academic [19] environments. These automatic analog/RF IC sizing methods aim to find the best set of device sizes by iterating over tentative guesses and evaluating their impact on circuit performance, as shown in Fig. 1.3. This process is shown to produce usable designs, but it is still slow, and reuse usually involves new optimization runs. Some approaches have been made to use ML algorithms in analog IC sizing. However, most of the different approaches incorporating ML/DL concepts in the simulation-based sizing loop attempted to alleviate the simulator workload by building models that estimate the circuit performances, fully bypassing simulation.

From the analysis of state-of-the-art sizing tools available for analog IC and RF design automation described in Chap. 2 of this book, simulation-based sizing optimization tools are widely accepted approaches as they keep the solutions' accuracy high and assist in the design of modern blocks. For instance, in IoT applications, where the demand for ultralow-power (ULP) radios is significant, voltage-controlled oscillators (VCOs) are extremely relevant design blocks. Besides requirements such as phase noise and power consumption, the intrinsic tradeoff specifications, such as the frequency tuning range and frequency pushing due to the supply voltage variation, must be carefully considered in a practical design. Building a realistic analysis of the design tradeoffs is a challenging task, as multiple tuning modes deliver a

Fig. 1.3 Simulation-based sizing concept



vast number of conflicting performance figures that need to be balanced. Adding further complexity, the impact of the process variations or parasitic layout structures turns the optimal balance of the design tradeoffs beyond human capabilities, which can only be solved with the assistance of automatic tools. Still, some challenges arise when designing complex RF circuit topologies using simulation-based sizing optimization:

- In some cases, the simulator is unable to converge to the guessed oscillation frequency, whereas in others, the simulation attempts to converge infinitely;
- The increase in simulation time of extracted netlists, as opposed to the pre-layout one, makes it harder to decide when to put a timeout on convergence attempts;
- An oscillation frequency guess must be provided for the steady state (SST) analysis. However, it is strongly correlated with the convergence of the analysis. Therefore, promising designs may still be lost without simulating multiple guesses.

1.4 Conclusion

While the demand for new analog ICs for ever-increasing challenging specifications keeps building momentum, their design automation is still lacking and needs to be addressed. In this book, we explore how ML/DL can be used to increase the effectiveness of analog and RF automatic sizing. Chapter 3 proposes a methodology that disrupts the most recent trials of replacing the simulator in the simulation-based sizing with ML/DL by proposing two distinct ANN models. The 1st classifies the convergence of the circuit for nominal and PVT corners, bypassing solutions that will hardly produce valuable information for the evolutionary kernel, and the 2nd predicts the VCOs' oscillating frequencies for the conditions above. The convergence classifier (CC_{ANN}) and frequency guess predictor (FGP_{ANN}) are seamlessly integrated into the simulation-based sizing loop, as shown in Fig. 1.1, accelerating and complementing the optimization process. And, in Chap. 4, a PVT regressor that inputs the circuit's sizing and the nominal performances to estimate the PVT corner performances via multiple parallel ANNs greatly accelerates robust design. Two control phases prevent the optimization process from being misled by inaccurate performance estimates. Ultimately speeding up the analog/RF IC optimization-based sizing concept, complementing the simulation process with artificial neural networks, and reducing the simulator workload.

References

1. Lourenço N, Martins R, Horta N (2017) Automatic analog IC sizing and optimization constrained with PVT corners and layout effects. Springer, Berlin

2. The Business Research Company (2020) Integrated circuits global market report 2020. Technical report
3. Gielen GGE, Rutenbar RA (2000) Computer-aided design of analog and mixed-signal integrated circuits. *Proc IEEE* 88(12):1825–1854
4. Martins R, Lourenço N, Horta N (2012) Generating analog IC layouts with LAYGEN II. Springer briefs in applied sciences and technology. Springer, Berlin
5. Cadence (2019) Virtuoso analog design environment GXL. Retrieved from <http://www.cadence.com>, Mar 2019
6. MunEDA (2019) WIKED™. Retrieved from <http://www.muneda.com>, Mar 2019
7. Afacan E, Lourenço N, Martins R, Dündar G (2021) Review: machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test. *Integr VLSI* 77:113–130
8. Suissa A et al (2010) Empirical method based on neural networks for analog power modeling. *IEEE TCAD* 29(5):839–844
9. Wolfe G, Vemuri R (2003) Extraction and use of neural network models in automated synthesis of operational amplifiers. *IEEE TCAD* 22(2):198–212
10. Alpaydin G, Balkir S, Dündar G (2003) An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Trans Evol Comput* 7(3):240–252. <https://doi.org/10.1109/TEVC.2003.808914>
11. Liu H, Singhee A, Rutenbar RA, Carley LR (2002) Remembrance of circuits past: macromodeling by data mining in large analog design spaces. In: *Proceedings 2002 design automation conference*, pp 437–442
12. Lourenço N et al (2019) Using polynomial regression and artificial neural networks for reusable analog IC sizing. In: *16th International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design*, pp 13–16, July 2019
13. Zhu K et al (2019) Genius route: a new analog routing paradigm using generative neural network guidance. In: *Proceedings of the ICCAD*
14. Guerra D, Canelas A, Póvoa R, Horta N, Lourenço N, Martins R (2019) Artificial neural networks as an alternative for automatic analog IC placement. In: *International conference on SMACD*, Lausanne, Switzerland, July 2019
15. Gusmão A, Passos F, Póvoa R, Horta N, Lourenço N, Martins R (2020) Semi-supervised artificial neural networks towards analog IC placement recommender. In: *IEEE International symposium on circuits and systems*, Seville, Spain, Oct 2020
16. Gusmão A, Horta N, Lourenço N, Martins R (2022) Scalable and order invariant analog integrated circuit placement with attention-based graph-to-sequence deep models. In: *Expert systems with applications*. Elsevier, Amsterdam
17. Gusmão A, Póvoa R, Horta N, Lourenço N, Martins R (2022) DeepPlacer: a custom integrated OpAmp placement tool using deep models. In: *Applied soft computing*, vol 115. Elsevier, Amsterdam, 108188
18. Andraud M, Stratigopoulos H, Simeu E (2016) One-shot non-intrusive calibration against process variations for analog/RF circuits. *IEEE TCAS-I Regul Pap* 63(11):2022–2035
19. Gonzalez-Echevarria R et al (2017) An automated design methodology of RF circuits by using pareto-optimal fronts of EM-simulated inductors. *IEEE Trans Comput Des Integr Circ Syst* 36(1):15–26

Chapter 2

Background and Related Work



2.1 Knowledge-Based Sizing

As shown in Fig. 2.1a [1], knowledge-based sizing tools, e.g., IDAC [2] and BLADES [3], have attempted to systematize the design by making use of a design plan obtained from expert knowledge. These tools solve a pre-designed plan using circuit equations and a hard-coded design strategy to build component sizes that meet the performance requirements. While this approach presented satisfactory results for automatic analog IC sizing, being its main advantage of the short execution time. However, deriving the design plan is complex and requires a high development time. Additionally, the continuous effort required to keep the design plan up to date with the advances in fabrication technologies, the increase in the complexity of circuit topologies, and the fact that the results obtained are not simulator-accurate make this approach suitable only as a first-cut design.

2.2 Optimization-Based Sizing

Aiming for optimality, the next generation of sizing tools has applied optimization techniques to analog/RF IC sizing, which can be further classified into equation-based or simulation-based when considering the method used to evaluate the circuit's performance, as illustrated in Fig. 2.1b [1].

2.2.1 Equation-Based Evaluation

Equation-based methods use analytic design equations to describe the circuit performance, and then, to resolve the degrees of freedom, tools such as OPASYN [4]

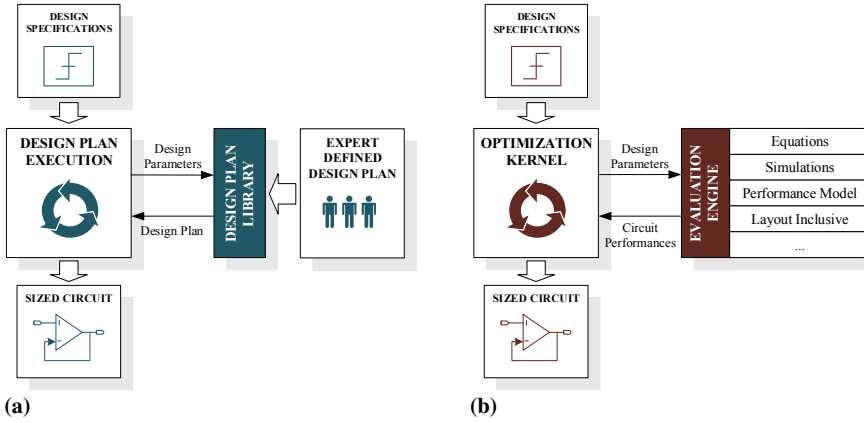


Fig. 2.1 Automatic circuit sizing methods: **a** knowledge-based and **b** optimization-based [1]

and CADICS [5] were used. Since the design equations had to be deduced by hand, the symbolic simulator ISAAC [6] was developed to automatically produce design equations used to evaluate the circuit performance in a relatively less complex process. This process reduces the setup time when optimizing new circuit topologies. Nonetheless, similar to knowledge-based sizing, the problem of using these methods is still mapping design characteristics by analytic equations, which is not straightforward. The approximations introduced in the equations result in poor accuracy when compared with the circuit simulator.

2.2.2 Simulation-Based Evaluation

Simulation-based optimization is the most prevalent method in both industrial [6, 7] and academic environments since designers prefer to avoid the risks of estimation errors in equation-based performance approximation. These simulation-based methods use an off-the-shelf circuit simulator to evaluate the circuit's performance. The main advantage of these approaches over the equation-based evaluation is its improved generalization capabilities, still, due to the long execution times required for some SPICE-based circuit evaluations and since a considerable number of simulations (hundreds to thousands) are required to reach the desired solutions, it may result in a time-consuming optimization process. One example of this approach was introduced in [8], where an analog IC design automation environment called AIDA implements a design flow from a circuit-level specification to a physical layout description. AIDA was a combination and integration of two in-house tools, GENOM-POF [9], responsible for the circuit synthesis and whose architecture is illustrated in Fig. 2.2, and, LAYGEN II [10], the AIDA's layout generator.

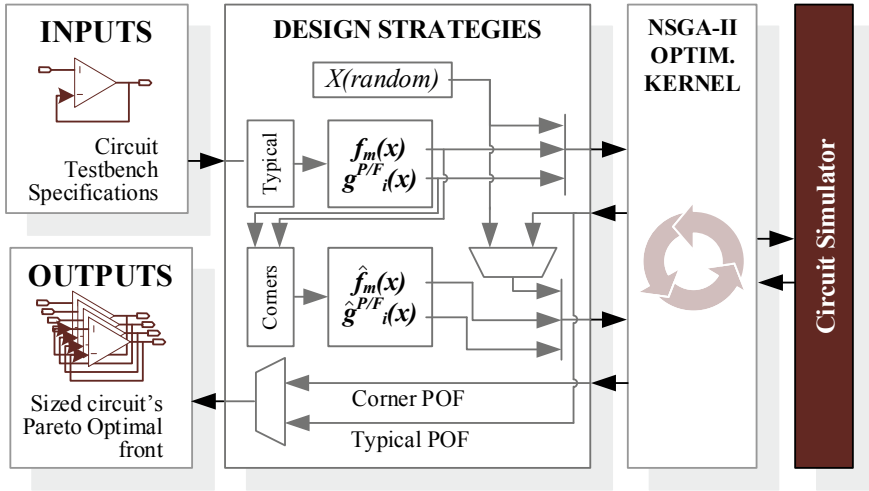


Fig. 2.2 GENOM-POF architecture based on the NSGA-II and using circuit simulator [8]

GENOM-POF was based on the multi-objective evolutionary optimization kernel NSGA-II [11] and used the industry standard circuit simulator Mentor Graphics' HSPICE as an evaluation engine to address robust design requirements, e.g., corner analysis. Corner analysis is one of the most used techniques for analog IC design centering. It corresponds to a worst-case approach in which a sizing solution for a given circuit topology is simulated over multiple combinations of parameter variations, such as process, power supply, or temperature. In GENOM-POF, the designer's inputs were the circuit and its testbench in the form of HSPICE netlist(s). These netlists must have, as parameters, the optimization variables and must include a method to measure the circuit's performance. The designer also had to define the desired range of the optimization variables, design constraints, and optimization objectives.

In the past few years, RF IC design has exploited this concept of simulation-based sizing [12–18]. Regarding AIDA-C [19], an enhanced version of GENOM-POF whose general flow is illustrated in Fig. 2.3, represents a generic number of candidate circuit sizing solutions, P , proposed by the optimization engine, where each one is a series of possible combinations of design variables. It is an iterative process, where in each iteration, the framework simulates the several test benches, K , affected by each sizing of P , to extract the desired measures. The most widely adopted commercial circuit simulators are supported in this process, such as Cadence's SPECTRE, Synopsys' ELDO, or Mentor Graphics' HSPICE. Due to a measure-processing interface, it is possible to combine measures from different test benches into composed expressions, which can be used as targets for the constrained multi-objective optimization problem.

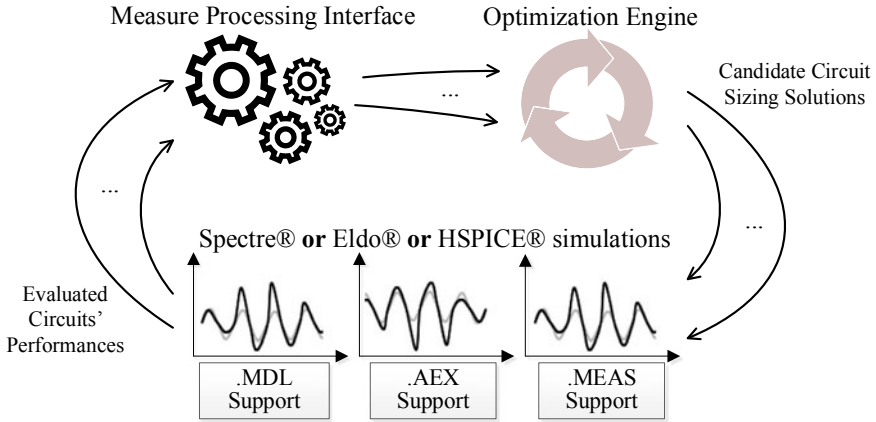


Fig. 2.3 Design flow of a multi-test bench analog and RF IC sizing optimization [19]

2.3 Machine Learning in Simulation-Based Evaluation

As each simulation made within an optimization-based loop may be a time-consuming process, an effort has been made to develop techniques that reduce the workload of the simulator, most of them focused on ML. ML is a subset of artificial intelligence, even though the latter aims to build complex decision systems and the former focuses on the statistical properties of data [20]. In his essay on probability theory, Thomas Bayes proposed several theorems [21] that laid the theoretical foundations for statistical learning and is the cornerstone for some early ML techniques, such as naive Bayes or Markov chains. In 1951, the first artificial neural machine was proposed. However, artificial neural networks (ANNs) only began to receive more attention from the community with Frank Rosenblatt's perceptron [22] and back-propagation [23] in 1958 and 1986, respectively, wherein the latter principles of dynamic programming were introduced. In the meantime, many other accomplishments have been achieved, and today there are a vast number of different ML techniques for solving classification and regression tasks.

In a classification problem, the main goal is to categorize data correctly. A typical example is the email spam filter which assigns incoming emails to the “spam” or “not-spam” categories. In a regression problem, the system's primary goal is to describe one or more continuous-value dependent variables as functions of the data observations. An example of regression is predicting house prices given the house's features (size, the number of rooms, location, etc.). A visual comparison between these two problems is shown in Fig. 2.4.

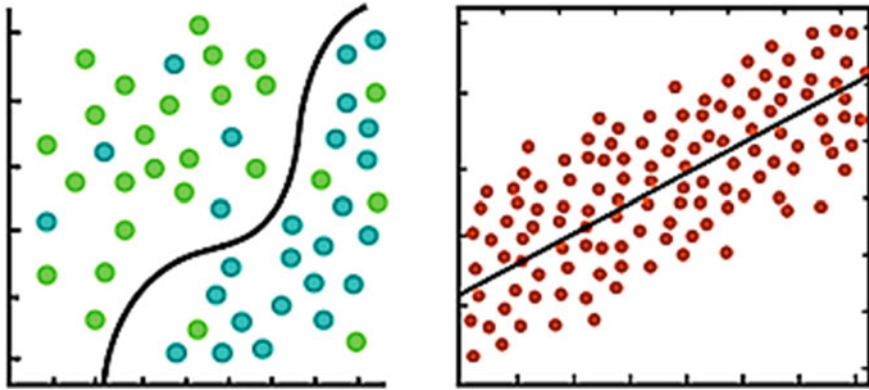


Fig. 2.4 Classification (left) versus regression (right) problems [25]

2.3.1 *Types of Supervision*

A critical characteristic of all ML systems is their ability to adapt correctly to new, previously unseen data and avoid overfitting the training data. Overfitting occurs when the system learns the detail and noise in the training data to the extent that it negatively impacts the performance of the system on new data, affecting its generalizing capabilities [20, 24]. Another attribute of an ML system is the amount and type of supervision. The following subsections refer to the different supervision categories in which an ML system can be categorized.

2.3.1.1 Supervised Learning

For these models, the dataset used must have some observations and the expected results, called labels. The labels can further divide this method into two problem-solving techniques: classification and regression. As stated, a classification problem is when the output variable is part of a group, for instance, “dog” or “not dog,” whereas regression is a case where the output variable is a real value, such as the cost of a house. Some examples of important supervised learning algorithms are linear regression, logistic regression, decision trees, support vector machines (SVMs), and ANNs.

2.3.1.2 Unsupervised Learning

The training uses data that has not been labeled for these models, aiming to create models that can draw inferences from datasets to describe hidden structures. Unsupervised learning is particularly suitable for problems such as clustering or association. A clustering problem is characterized by disclosing the inherent groupings in

the data, for instance, grouping people based on their age, whereas an association problem aims to discover relations rules within the data, such as people who buy X also tend to buy Y. Some examples of relevant algorithms are principal component analysis, K-means, and mixture models.

2.3.1.3 Semi-supervised Learning

Semi-supervised learning is the third category that falls between supervised learning and unsupervised learning. Its training data combines a small amount of labeled data with a large amount of unlabeled data, and the system is trained with a combination of supervised and unsupervised algorithms. Some algorithm examples are autoencoders and deep belief networks.

2.3.2 *Simulation-Based Sizing Enhanced with SVMs*

With the recent technology advancements, the use of macro models such as ANNs or SVMs introduced another type of optimization-based sizing, designated numerical-model based. Given the models' prediction speed, by using these tools, one can reduce the high simulation times of the simulator loop by aiding the simulator in certain tasks or, in a more drastic way, by completely replacing the simulator.

Starting with SVMs, which is a supervised learning algorithm for data separation, making use of linear combinations to produce a boundary that maximizes the margin between classes. This algorithm is especially interesting if the data is linearly separable, as for nonlinear patterns, a kernel trick is used, allowing the SVM to create this boundary in a higher dimension hyperplane.

In [1], an SVM classifier is used to enhance the multi-objective IC sizing optimization process. The previously presented analog IC sizing tool, GENOM-POF, was used to demonstrate the methodology. The SVM is used to create feasibility models that diminish the design search space during the optimization process, reducing the number of required evaluations. This approach was validated using benchmark examples consisting of two different circuits, a single-ended folded cascode amplifier, and a fully differential telescopic amplifier. The functional feasibility regions used to train the feasibility model were defined by functional constraints, where the training data used to train the model was obtained using a design of experiments. The sampled points obtained were sorted into 3 classes, feasible, quasi-feasible, and infeasible. Finally, the evaluation was made, where the model classifies the individuals based on their classes, discarding the unfeasible ones. The results showed that the models had absolute gains ranging from 10 to 20% in terms of the overall reduction in the number of evaluations required. As the electrical simulation is more time-consuming than the SVM model evaluation, it allows an efficient diminishing of the design search space.

In [26], an SVM is also used to identify the feasibility of the design space of analog circuits to reduce a large amount of the entire design space, sampling only the points considered feasible and their neighbors. After choosing the proper parameters of the SVM, the resulting model may reach 100% accuracy on the training data. So, the difficulty relies on the generalization capability when facing an independent validation data set. To tackle this issue, three accuracy metrics were presented, i.e., overall accuracy, percentage of false negatives, and percentage of false positives. These metrics are presented in Eqs. (2.1), (2.2), and (2.3), respectively:

$$P_t = \frac{\text{Number of correctly classified samples}}{\text{Number of samples in the validation set}} \quad (2.1)$$

$$P_{fn} = \frac{\text{Number of false negatives}}{\text{Number of positives in the validation set}} \quad (2.2)$$

$$P_{fp} = \frac{\text{Number of false positives}}{\text{Number of predicted positives}} \quad (2.3)$$

Two circuits were used as a case study to validate this method: an operational transconductance amplifier and a low-voltage double-balanced mixer. With discarding the predicted negative values, the coverage of the feasibility design space was consistently above 99% for both circuits, and the rate of feasible designs that were excluded from being sampled was in the order of 10^{-4} . Finally, the computational time was also reduced, where the results show between 59 and 71% reduction when compared with previous approaches. The problem with using SVMs is that tuning hyper-parameters and selecting the correct kernel is quite challenging. SVMs also have degraded performance when faced with large datasets and a higher number of features.

2.3.3 *Simulation-Based Sizing Enhanced with ANNs*

Today, ANNs are pretty popular in ML due to the increased data and computing power available. These two factors prevented researchers from using them altogether in academic settings in the past. However, nowadays, with fast-processing computers, ANNs can be found in image processing, speech recognition, and other areas where large amounts of data are available. They are systems based on the human brain, copying how we learn and make decisions. These networks are composed of an input and an output layer and one or more hidden layers. Each of these layers is a combination of neurons, where the input layer is where the data is fed to the network and the output layer is where the algorithm results are obtained.

The advantages of using ANNs are their high performance, capability to solve problems impossible for humans, an excellent algorithm for regression and classification problems, and ability to handle large amounts of data. Some disadvantages

are their black-box nature (i.e., it is difficult for researchers to completely understand why the algorithm behaves a certain way because of how the numerical values are produced), the long time to train the model, and the large amount of data required. Nonetheless, ANNs have been proven to build effective end-to-end ML systems, which can also be used for the design automation of analog and RF ICs [27–29]. These may span from modeling [30], synthesis [31], layout generation [32–37], or even fault testing [38].

In [39], a neural network-based methodology is used to estimate the performance parameters of different CMOS operational amplifier topologies. To obtain the efficiency and accuracy of the resulting performance models, these were used in a genetic algorithm-based circuit synthesis system. The performance parameters of the synthesized circuits were validated by SPICE simulations and compared with the ones predicted by the ANN models. Training data of the model was directly generated through SPICE simulations to provide accurate and reliable data to the system. The ANN's architecture had only one hidden layer, with its number of neurons ranging from 8 to 13. However, its generalization capabilities were limited, as in some performances predicted by the model, the test error reached 60%.

This approach proved to be much faster when compared to the traditional SPICE simulation. The genetic algorithm, using the ANN models, was executed 10,000 times to produce 8 performance parameters, each obtained through a different iteration. Through SPICE simulation, each of these iterations would require 2 s to complete, which would total about 44 h ($10,000 \times 2 \times 8$) for all configurations. The execution time using ANN models was about 80 s for all configurations, representing a speed-up factor of $2000\times$. Beyond some of the higher reported errors, the models also proved to be capable of capturing nonlinear behavior of the performance characteristics of a circuit which requires a large number of simulations, but in the end, the effort is justified when considering the reusability of the models in other amplifier topologies.

In [40], an ANN with two hidden layers is used to replace the SPICE simulator. Multi-objective optimization is frequently used in analog sizing to reveal the trade-offs of the design specifications with the help of Pareto optimal fronts (POFs). A rough POF can be found in a reasonable time with multi-objective optimization, but the high-quality ones require a large amount of simulator iterations, resulting in long synthesis periods. In this paper, a method is presented to speed-up this process. After a multi-objective optimization phase to obtain a, designated, low-quality POF, the process switches to a faster single-objective optimization to complete the POF making it smoother and more continuous. At this phase, the SPICE simulator was also replaced by an ANN, reducing the synthesis time even further. The training data for the ANN was obtained from the multi-objective optimization phase.

This method was applied to the design of two circuits, a two-stage amplifier, and a folded cascode operational transconductance amplifier to validate this tool. For the first circuit a speed-up factor of about $29.7\times$ was obtained, which translates to a 96.6% time reduction, with a maximum error of 0.44%. As for the second circuit, a speed-up factor of 28.3 was obtained, representing a 96.4% time reduction, with a maximum error of 1.55%.

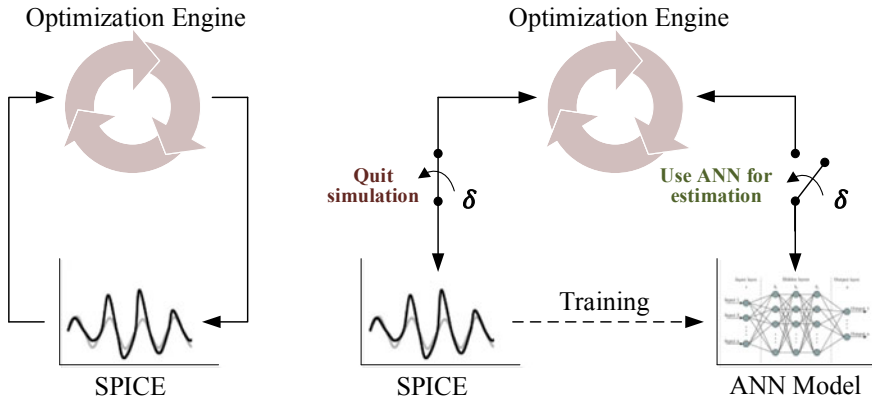


Fig. 2.5 **a** General flow of simulation-based synthesizer and **b** modified flow with ANN as performance estimator [40]

In [41], a similar method is used to accelerate a simulation-based circuit synthesizer using ANNs to determine circuit performances instead of a SPICE simulator. Instead of training the ANN with simulation data beforehand and simply replacing the simulator with the trained ANN, the simulation-based synthesizer is left unchanged for some generations of the optimization loop, and only after a period the ANN replaces the SPICE simulator. Unlike other conventional algorithms, all the data generated in the first phase is used as training data for the ANN instead of being discarded. The proposed circuit synthesizer flow is shown in Fig. 2.5. Since the training data acquisition step consumes a significant amount of time and is necessary for every new topology to be treated, the main innovation of this approach is that no separate data acquisition step to train the ANNs is required. Therefore, the flow can be used for every new topology without losing generality for all analog circuits.

This method was applied to the circuit synthesis phase of two circuits, a single-stage amplifier, and a folded cascode operational transconductance amplifier. With only the SPICE simulator, the circuit sizing of the single-stage amplifier took 4.92 h to complete, where the optimizer was executed for 100 generations. With ANNs replacing the simulator after the first 20 generations, the execution time was reduced by 64.8%, corresponding to a speed factor of $2.84\times$, with errors below 1%. For the folded cascode amplifier, which presents a higher complexity, the original optimizer took 400 generations and 22.58 h to complete the circuit sizing. The best time reduction obtained with ANNs was 50.3% with errors below 1%, where the ANN replaced the simulator after 155 generations.

All these approaches to reduce the execution time of optimization-based sizing use ANNs to replace or complement the circuit simulator. The execution time is greatly reduced by avoiding time-consuming circuit simulations. However, in [39, 42], at later stages of the optimization, the circuit simulator is re-established to recover the accuracy lost. Furthermore, the ANN models are trained over the entire design space, which spends valuable resources modeling and evaluating large regions of unusable

Table 2.1 Speeding-up simulation-based sizing with ANNs overview

Reference	Speedup factor	Maximum error	Number of layers	Method
Wolfe and Vemuri [39]	≈ 2000	60%	3	Complement/Replace simulator
Çakıcı et al. [40]	29.7	1.55%	3	Replace simulator
İslamoğlu et al. [41]	2.8	0.77%	4 and 5	Semi-replace simulator
Hakhamaneshi et al. [44]	n/r	n/r	4	Replace simulator
Alpaydin et al. [42]	n/r	n/r	3	Complement simulator
Liu et al. [43]	n/r	$\approx 10,000\%$	3	Replace simulator

n/r—not reported

design combinations. In [43], the ANNs were also trained to replace the simulator, but the previous issue is somewhat addressed by applying data mining techniques to build a model that captures only significant regions of the performance space visited during automatic synthesis.

In [44], deep neural networks (DNNs) boost the optimizer's sample efficiency. With the use of an oracle, a comparison is made between two designs, in terms of each design variable, as a method to select which of the two designs is likely to have better performance figures in advance. Since DNNs are especially good at approximating complex functions and have a good generalization to unseen samples, a DNN model is derived to behave as an oracle, which is in fact a simulator. This discriminator achieves at least two orders of magnitude in sample efficiency, representing a considerable reduction in the number of simulations required. A summary of these tools is shown in Table 2.1.

2.4 Other ML/DL Efforts on Analog/RF Sizing

This section overviews different ML applications in the analog/RF IC sizing domain.

2.4.1 Predicting Sizing from Performances

Using ANNs to find device sizing in analog IC proved to be feasible. These methods learn and output a candidate circuit sizing when asked for target specifications [45, 46]. In [31], an ANN is developed to give the channel widths of all the transistors in a circuit when the designer gives the desired output specifications. The training phase data was performed with different SPICE parameters from the ones used in the test data in order to show the ability to give the transistor sizes of a circuit

for new untrained technology, having no dependency on the SPICE parameters. As a method of validation, two circuits were used, current mirrors and a CMOS differential amplifier. For the first one, a general regression neural network was used, and the results showed that it could estimate the current mirror's transistor sizes for new technologies with 94% accuracy. For the second one, a multilayer perceptron was used, and its results had an accuracy of 90%.

In [47], to produce the sizing for a low noise amplifier, several ANNs are put in sequential order, having as input the intended performance. The results have shown good prediction accuracy. However, the training and tuning of such a model have proven to be exceedingly tricky. While only 277 handmade sizing solutions were used for the training phase, an outer loop was still required to define the model's hyperparameters, reflected in a train of over 5 h on such a short dataset. In [48], the sizing for an amplifier is also predicted using ANNs when given its specifications. However, in this work, the model and training phases are different since the test was only performed on ten samples from the original dataset, and no evaluation is made on the performance and usability of the model for unknown target specifications.

2.4.2 Reinforcement Learning

Reinforcement learning (RL) aims to develop an agent that learns how to behave in a particular environment where the only feedback is the reward of its actions. This interaction between the agent and environment is depicted in Fig. 2.6. The agent's primary goal is to maximize the notion of cumulative reward regarding its actions. These systems can teach robots to learn motor skills [49] or master complex board games like chess or Go [50].

RL can also be used in analog/RF sizing. In [51], an agent learns from trial and error how to behave like a circuit designer evolving to discover circuit sizes that satisfy the performance specifications finally. Another instance where RL is used for sizing is in [52], a tool named AutoCkt, an ML optimization framework trained using deep RL, that is capable of finding post-layout circuit parameters for a particular parameter

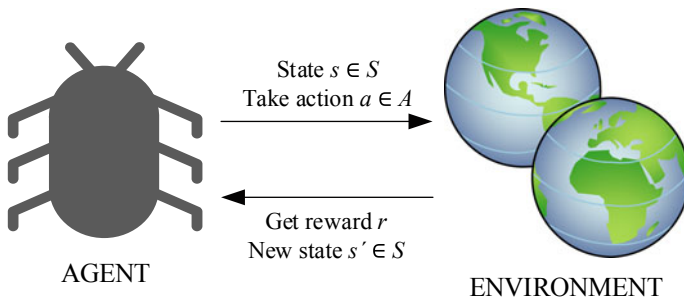


Fig. 2.6 Interaction between agent and environment

specification and can also acquire knowledge about the entire circuit design space using a sparse subsampling technique.

This method uses models to make a sequence of decisions. The agent observes and interacts with an uncertain, potentially complex environment by selecting and executing actions, following a trial-and-error approach, and getting rewards or penalties according to what action it performs. The agent is trained to learn a policy that maximizes the expected outcome of the actions over time. These methods have been used to play complex board games or, for instance, in an autonomous vehicle to put safety first or minimize ride time. This method has been used in alternating current optimization in [51–53]. With deep learning, an agent is trained, not needing previous knowledge about optimizing circuits.

2.5 Case Study

To justify the motivation behind the developments proposed in this book, and to demonstrate how a time-consuming simulation step may hinder an effective optimization-based synthesis, a case study is introduced within this section, which will be explored in the following chapters.

2.5.1 *Dual-Mode Class C/D VCO*

The development of the proposed methodologies will be tested on the sizing of a complex dual-mode class C/D voltage-controlled oscillator (VCO), whose schematic is presented in Fig. 2.7 and whose optimization was firstly described in [54]. In that work, instead of achieving the desired performance parameters with sequential single-objective optimizations, a single many-objective sizing optimization, described as “everything-at-once” optimization, is proposed to find the best performance boundaries. The circuit simulator performed a multi-process corner analysis and the optimization followed a worst-case corner criteria on top of a worst-case tuning range optimization, taking into account two different tuning modes, b_{0000} and b_{1111} . The results pushed the circuit to its performance limits, reducing to almost half of the power consumption of the original design, and showing its potential for ultralow-power with more than 93% reduction. In the optimizations carried, there were 28 optimization variables that affected the sizing of 43 devices. The full list can be found in Table 2.2.

A total of 18 performance figures were considered from 7 different testbenches and the optimizations were performed with populations of 512 elements optimized through 200 generations only. Each optimization took approximately 50 h to complete in an Intel-Xeon CPU E5-2630-v3@2.40 GHz with 64 GB of RAM workstation using eight cores for parallel evaluation, i.e., more than 2 days. Nonetheless, a complete process, voltage and temperature (PVT) corners optimization of this circuit is desired,

Table 2.3 List of TT and PVT testbenches for the dual-mode class C/D VCO

Name	Process	Voltage (V)	Temperature (°C)
Typical (TT)	TT	0.35	25
Fast–Fast (FF)	FF	0.35	25
Fast–Slow (FS)	FS	0.35	25
Slow–Fast (SF)	SF	0.35	25
Slow–Slow (SS)	SS	0.35	25
300mV	TT	0.3	25
400mV	TT	0.4	25
m40dC	TT	0.35	–40
85dC	TT	0.35	85

Table 2.4 List of performances considered for the dual-mode class C/D VCO in TT and PVT corners

Measure	Units	Description
f_{osc}	GHz	Oscillation frequency
PN@10 kHz	dBc/Hz	Phase noise at 10 kHz
PN@100 kHz	dBc/Hz	Phase noise at 100 kHz
PN@1 MHz	dBc/Hz	Phase noise at 1 MHz
PN@10 MHz	dBc/Hz	Phase noise at 10 MHz
Power	mW	Power consumption
FOM@10 kHz	dBc/Hz	Figure-of-merit at 10 kHz
FOM@100 kHz	dBc/Hz	Figure-of-merit at 100 kHz
FOM@1 MHz	dBc/Hz	Figure-of-merit at 1 MHz
FOM@10 MHz	dBc/Hz	Figure-of-merit at 10 MHz

performance figures. The full list of testbench variations can be seen in Table 2.3 and the list of performances in Table 2.4.

2.5.2 Dataset Generation

To generate the dataset an optimization was conducted in order to minimize both power and phase noise at 10 MHz in both tuning modes, while imposing constraints on other 7 measured performances for each testbench, i.e., in all combinations of PVT corners and tuning modes. These optimization constraints and objectives are shown in Table 2.5. The optimization was executed through a total of 350 generations and took a total of 612 h to complete and resulted in a total of 27 sizing solutions. In Fig. 2.8 is shown the POF evolution of the original optimization which contains the best sizing solutions throughout the generations, and, in Table 2.6 the values of the final POF obtained at generation 350 are shown.

Table 2.5 Optimization constraints and objectives

Tuning mode	Measure	Testbenches	Units	Opt. constraint	Opt. objective
<i>b</i> ₀₀₀₀	<i>f</i> _{osc}	All	GHz	≥ 4.8	
	PN@ 10 kHz	All	dBc/Hz	≤ − 49	
	PN@ 100 kHz	All	dBc/Hz	≤ − 76	
	PN@ 1 MHz	All	dBc/Hz	≤ − 98	
	PN@ 10 MHz	All	dBc/Hz	≤ − 119	Minimize
	Power	All	mW	n/d	Minimize
	FOM@ 10 MHz	All	dBc/Hz	≥ − 180	
<i>b</i> ₁₁₁₁	<i>f</i> _{osc}	All	GHz	≤ 3.9	
	PN@ 10 kHz	All	dBc/Hz	≤ − 55	
	PN@ 100 kHz	All	dBc/Hz	≤ − 82	
	PN@ 1 MHz	All	dBc/Hz	≤ − 103	
	PN@ 10 MHz	All	dBc/Hz	≤ − 124	Minimize
	Power	All	mW	n/d	Minimize
	FOM@ 10 MHz	All	dBc/Hz	≥ − 180	

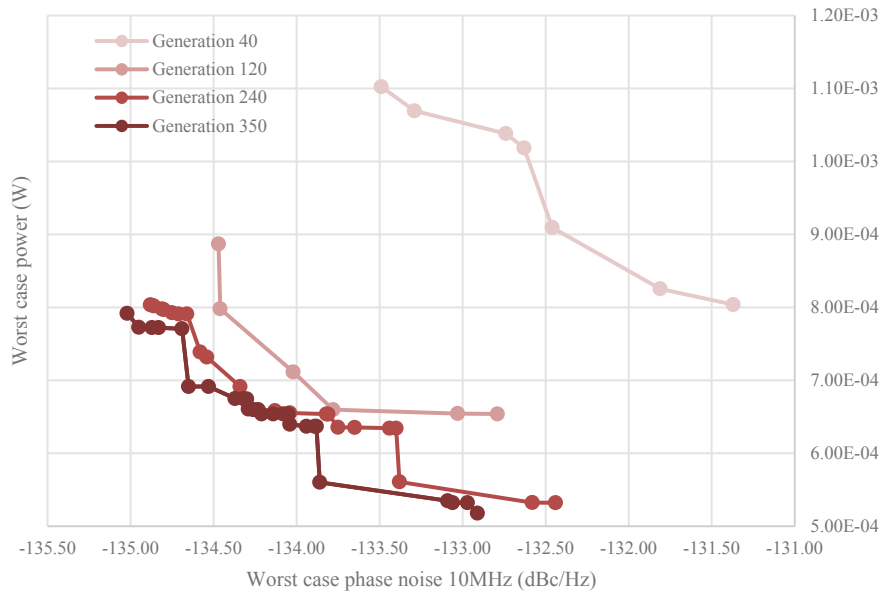


Fig. 2.8 POF evolution throughout the original optimization

Table 2.6 Final solutions of the original POF at generation 350

Worst case phase noise 10 MHz (dBc/Hz)	Worst case power (mW)	Worst case phase noise 10 MHz (dBc/Hz)	Worst case power (mW)
− 135.02	7.918×10^{-1}	− 134.14	6.539×10^{-1}
− 134.95	7.727×10^{-1}	− 134.08	6.537×10^{-1}
− 134.87	7.722×10^{-1}	− 134.06	6.537×10^{-1}
− 134.83	7.722×10^{-1}	− 134.05	6.536×10^{-1}
− 134.69	7.707×10^{-1}	− 134.04	6.397×10^{-1}
− 134.65	6.914×10^{-1}	− 133.94	6.369×10^{-1}
− 134.53	6.914×10^{-1}	− 133.89	6.369×10^{-1}
− 134.37	6.749×10^{-1}	− 133.88	6.368×10^{-1}
− 134.32	6.746×10^{-1}	− 133.86	5.601×10^{-1}
− 134.30	6.746×10^{-1}	− 133.09	5.348×10^{-1}
− 134.29	6.605×10^{-1}	− 133.06	5.323×10^{-1}
− 134.26	6.598×10^{-1}	− 132.97	5.322×10^{-1}
− 134.23	6.596×10^{-1}	− 132.91	5.178×10^{-1}
− 134.21	6.539×10^{-1}		

2.6 Conclusion

In this chapter, different methods used in circuit optimization sizing were introduced and compared regarding the evaluation engine used, emphasizing simulation-based sizing. Using a simulator as the evaluation engine is the most widely accepted approach, with its main advantages being its generality and easy-and-accurate model. The main problem with this method is how time-consuming it may become when the SPICE-like simulation times increase. In recent works, the use of ML tries to address and solve this problem by often introducing SVM and ANNs into the optimization phase by either replacing or complementing the circuit simulator.

References

1. Lourenço N, Martins R, Barros M, Horta N (2013) Analog circuit design based on robust POFs using an enhanced MOEA with SVM models. In: Fakhfakh M, Tlelo-Cuautle E, Castro-Lopez R (eds) Analog/RF and mixed-signal circuit systematic design. Lecture notes in electrical engineering, vol 233. Springer, Berlin
2. Degrauwe M et al (1987) IDAC: an interactive design tool for analog CMOS circuits. IEEE J Solid-State Circ 22(6):1106–1116
3. El-Turky F, Perry EE (1989) BLADES: an artificial intelligence approach to analog circuit design. IEEE Trans Comput Aided Des Integr Circ Syst 8(6):680–692
4. Koh H, Sequin CH, Gray PR (1990) OPASYN: a compiler for CMOS operational amplifiers. IEEE Trans Comput Aided Des Integr Circ Syst 9(2):113–125

5. Jusuf G, Gray P, Sangiovanni-Vincentelli A (1990) CADICS—cyclic analog-to-digital converter synthesis. In: Proceedings of ACM/IEEE ICCAD, pp 286–289
6. Gielen G, Walscherts H, Sansen W (1989) ISAAC: a symbolic simulator for analog integrated circuits. *IEEE J Solid-State Circ* 24(6):1587–1597
7. Cadence (2019) Virtuoso analog design environment GXL. [Online]. Available: <http://www.cadence.com>. Accessed: 15 May 2019
8. Martins R, Lourenço N, Rodrigues S, Guilherme J, Horta N (2012) AIDA: automated analog IC design flow from circuit level to layout. In: International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design (SMACD), Seville, Sept 2012
9. Lourenço N, Horta N (2012) GENOM-POF: multi-objective evolutionary synthesis of analog ICs with corners validation. In: Genetic and evolutionary computation conference, Philadelphia, USA, July 2012
10. Martins R, Lourenço N, Horta N (2012) Generating analog IC layouts with LAYGEN II. Springer briefs in applied science and technology. Springer, Berlin
11. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6:182–197
12. Liu B, Deferm N, Zhao D, Reyaert P, Gielen G (2012) An efficient high-frequency linear RF amplifier synthesis method based on evolutionary computation and machine learning techniques. *IEEE TCAD Integr Circ Syst* 31(7):981–993
13. Póvoa R et al (2014) LC-VCO automatic synthesis using multiobjective evolutionary techniques. In: IEEE International symposium on circuits and systems, pp 293–296, June 2014
14. Afacan E, Dündar G (2016) A mixed domain sizing approach for RF circuit synthesis. In: IEEE International symposium on design and diagnostics of electronic circuits and systems, pp 1–4, June 2016
15. González-Echevarría R et al (2017) An automated design methodology of RF circuits by using pareto-optimal fronts of EM-simulated inductors. *IEEE Trans Comput Aided Des Integr Circ Syst* 36(1):15–26
16. Afacan E, Dündar G (2018) Design space exploration of CMOS cross-coupled LC oscillators via RF circuit synthesis. In: 15th International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design, pp 1–4, July 2018
17. Enhanced systematic design of a voltage controlled oscillator using a two-step optimization methodology
18. Afacan E, Dündar G (2019) A comprehensive analysis on differential cross-coupled CMOS LC oscillators via multiobjective optimization. *Integr VLSI* 67:162–169
19. Martins R et al (2020) Design of a 4.2-to-5.1 GHz ultralow-power complementary class-B/C hybrid-mode VCO in 65-nm CMOS fully supported by EDA tools. *IEEE Trans Circ Syst I Regul Pap* 67(11):3965–3977
20. Murphy KP (2012) Machine learning: a probabilistic perspective (adaptive computation and machine learning series). MIT Press
21. Bayes M, Price M (1763) An essay towards solving a problem in the doctrine of chances. By the late rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton. In: *Philosophical transactions* (1683–1775), vol 53, pp 370–418
22. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 65(6):386–408
23. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323(6088):533–536
24. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer, Berlin
25. Javatpoint. Accessed: Oct. 12, 2021. [Online]. Available: <https://www.javatpoint.com/regression-vs-classification-in-machine-learning>
26. Ding M, Vemur R (2005) An active learning scheme using support vector machines for analog circuit feasibility classification. In: International conference on VLSI design, Kolkata, India

27. Afacan E, Lourenço N, Martins R, Dündar G (2021) Review: machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test. *Integr VLSI* 77:113–130
28. Fayazi M, Colter Z, Afshari E, Dreslinski R (2021) Applications of artificial intelligence on the modeling and optimization for analog and mixed-signal circuits: a review. *IEEE TCAS-I* 68(6):2418–2431
29. Mina R, Jabbour C, Sakr G (2022) A review of machine learning techniques in analog integrated circuit design automation. *Electronics* 11(3):435
30. Suissa A et al (2010) Empirical method based on neural networks for analog power modeling. *IEEE Trans Comput Aided Des Integr Circ Syst* 29(5):839–844
31. Kahraman N, Yildirim T (2008) Technology independent circuit sizing for fundamental analog circuits using artificial neural networks. In: 2008 PhD research in microelectronics and electronics (PRIME). IEEE, pp 1–4
32. Zhu K et al (2019) Genius route: a new analog routing paradigm using generative neural network guidance. In: Proceedings of international conference on computer aided design (ICCAD)
33. Guerra D, Canelas A, Póvoa R, Horta N, Lourenço N, Martins R (2019) Artificial neural networks as an alternative for automatic analog IC placement. In: International conference on SMACD, Lausanne, Switzerland, July 2019
34. Gusmão A, Passos F, Póvoa R, Horta N, Lourenço N, Martins R (2020) Semi-supervised artificial neural networks towards analog IC placement recommender. In: IEEE International symposium on circuits and systems, Seville, Spain, Oct 2020
35. Gusmão A, Horta N, Lourenço N, Martins R (2022) Scalable and order invariant analog integrated circuit placement with attention-based graph-to-sequence deep models. In: Expert systems with applications. Elsevier, Amsterdam
36. Gusmão A, Póvoa R, Horta N, Lourenço N, Martins R (2022) DeepPlacer: a custom integrated OpAmp placement tool using deep models. In: Applied soft computing, vol 115. Elsevier, Amsterdam, 108188
37. Gusmão A, Horta N, Lourenço N, Martins R (2021) Late breaking results: attention in Graph2Seq neural networks towards push-button analog IC placement. In: ACM/IEEE design automation conference (DAC), San Francisco, USA, Dec 2021
38. Andraud M, Stratigopoulos H, Simeu E (2016) One-shot non-intrusive calibration against process variations for analog/RF circuits. *IEEE Trans Circ Syst I Regul Pap* 63(11):2022–2035
39. Wolfe G, Vemuri R (2003) Extraction and use of neural network models in automated synthesis of operational amplifiers. *IEEE Trans Comput Aided Des Integr Circ Syst* 22(2):198–212
40. Çakıcı TO, İslamoğlu G, Güzelhan ŞN, Afacan E, Dündar G (2020) Improving POF quality in multi objective optimization of analog ICs via deep learning. In: 2020 European conference on circuit theory and design (ECCTD), Sofia, Bulgaria, pp 1–4
41. İslamoğlu G, Çakıcı TO, Afacan E, Dündar G (2019) Artificial neural network assisted analog IC sizing tool. In: 2019 16th International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design (SMACD), Lausanne, Switzerland, pp 9–12
42. Alpaydin G, Balkir S, Dündar G (2003) An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Trans Evol Comput* 7(3):240–252
43. Liu H, Singhee A, Rutenbar RA, Carley LR (2002) Remembrance of circuits past: macromodeling by data mining in large analog design spaces. In: Proceedings 2002 design automation conference (IEEE Cat. No.02CH37324), New Orleans, LA, USA, pp 437–442
44. Hakhamaneshi K, Werblun N, Abbeel P, Stojanović V (2019) BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks. In: 2019 IEEE/ACM international conference on computer-aided design (ICCAD), Westminster, CO, USA, pp 1–8
45. Lourenço N et al (2018) On the exploration of promising analog IC designs via artificial neural networks. In: 2018 15th International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design (SMACD), Prague, pp 133–136
46. Lourenço N et al (2019) Using polynomial regression and artificial neural networks for reusable analog IC sizing. In: 16th International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design (SMACD). IEEE, pp 13–16

47. Dumesnil E, Nabki F, Boukadoum M (2015) RF-LNA circuit synthesis using an array of artificial neural networks with constrained inputs. In: 2015 IEEE International symposium on circuits and systems (ISCAS), Lisbon, pp 573–576
48. Takai N, Fukuda M (2017) Prediction of element values of OPamp for required specifications utilizing deep learning. In: 2017 International symposium on electronics and smart devices (ISESD), Yogyakarta, pp 300–303
49. Peters J, Schaal S (2008) Reinforcement learning of motor skills with policy gradients. *Neural Netw* 21(4):682–697
50. Silver D et al (2017) Mastering the game of go without human knowledge. *Nature* 550(7676):354–359
51. Zhao Z, Zhang L (2020) Deep reinforcement learning for analog circuit sizing. In: 2020 IEEE International symposium on circuits and systems (ISCAS), Sevilla. IEEE, pp 1–5
52. Settaluri K, Haj-Ali A, Huang Q, Hakhamaneshi K, Nikolic B (202) AutoCkt: deep reinforcement learning of analog circuit designs. In: 2020 Design, automation & test in Europe conference & exhibition (DATE), Grenoble, France, pp 490–495
53. Wang H et al (2018) Learning to design circuits. arXiv preprint [arXiv:1812.02734](https://arxiv.org/abs/1812.02734)
54. Martins R et al (2019) Many-objective sizing optimization of a class-C/D VCO for ultralow-power IoT and ultralow phase-noise cellular applications. *IEEE TVLSI* 27(1):69–82

Chapter 3

Convergence Classifier and Frequency Guess Predictor Based on ANNs



3.1 Contributions

Most of the different approaches incorporating machine learning (ML)/DL concepts in the simulation-based sizing loop attempted to alleviate the simulator workload by building models that estimate the circuit performances, bypassing simulation [8–10]. While an optimization using an ANN performance predictor can be $2000\times$ faster than one using the simulator, the errors reported for some performance space regions can reach 60% deviation [8]. These performance estimation models can be trained offline, or online [19, 20], where the data gathered from the current simulation-based process is used for model training. Once a satisfactory deviation error is reached, the ANN replaces the simulator. Without attempting to predict performances, only recently were ANNs used as an oracle [21]. When given two different circuit sizing solutions, the ANNs predict which design will perform better post-layout, avoiding unnecessary simulations of, in theory, inferior solutions.

Real-world applications demand robust VCOs, and to ensure it. Their sizing has moved towards exhaustive PVT or layout-inclusive optimizations. Still, convergence problems arise from these time-consuming simulations, and setting a timeout on the convergence attempt is a delicate decision. Additionally, the probability of convergence is strongly correlated with the *guess* oscillating frequency, which is always fixed on the frequency of interest during automatic sizing procedures [1–5]. Therefore, the major contributions of the work described in this chapter are listed as follows:

- Previous works [8–10, 19, 20] focus on simulator replacement on automatic sizing procedures. Even if the estimated performances present acceptable deviations, simulator-grade accuracy is lost. In this work, the proposed ANNs are used as a seamless filter/helper for exhaustive PVT-inclusive RF sizing optimization problems, and thus, the simulator is kept through the whole process;

- The convergence classifier ANN captures underlying relations between the sizes of the different components and (un)successful convergence attempts in nominal and PVT corners. Therefore, the model is capable of operating in different regions of the performance space and be reused for optimizations with a different set of targets, i.e., generalizing beyond training data;
- In [1–5], good VCO designs, i.e., feasible solutions or unfeasible but relevant information for the optimization process, are lost by simulating only one *guess*. The frequency guess predictor ANN provides an accurate prediction of which frequency a given combination of device dimensions is likely to oscillate, thus taking advantage of information that is often lost.

3.2 Classifier and Regressor Based on Deep ANNs

The goal is to reduce the total effort of the evaluation engine, i.e., the simulator, by reducing the number of simulated candidate circuit sizing solutions focusing on VCO circuits. A vital property of the VCO is that ill-designed circuits will not oscillate, leading to no convergence in the simulation. The combination of the proposed models will try to predict whether a particular solution can generate all the performance metrics, i.e., if it is likely that it will, at least, simulate, and when this is the case, at what frequency the circuit is likely to oscillate. These are critical factors in increasing the efficiency of state-of-the-art simulation techniques for oscillating circuits [22], which can waste immense resources trying to get some simulation output when the circuits do not oscillate. The optimization process is described in Fig. 3.1, where the DNNs, a classifier and a regressor, play the filter role of selecting from the candidate circuits that are most likely to be useful to the optimization process and, thus, to be presented to the simulator.

3.2.1 Underlying Architectures

The two ANNs that are going to be developed have similar architectures, with some minor differences. The convergence classifier ANN will have n input neurons, corresponding to the circuit sizing and the tuning mode (referent to the case study introduced in Chap. 2), and y output neurons for each PVT corner considered, as shown in Fig. 3.2. With these inputs, the ANN can predict whether a given candidate circuit sizing solution should be simulated or not, as the output will be a classification of “convergent” or “nonconvergent” for each possible candidate circuit sizing solution.

Besides having the same inputs as the classification model, the regression ANN has one additional input. The PVT corner is to be considered. The output will determine the oscillatory frequency for that specific corner and tuning mode, as shown in Fig. 3.3. This regression will be executed only for the solutions classified as “converge” since there is no point in predicting the oscillatory frequency for solutions

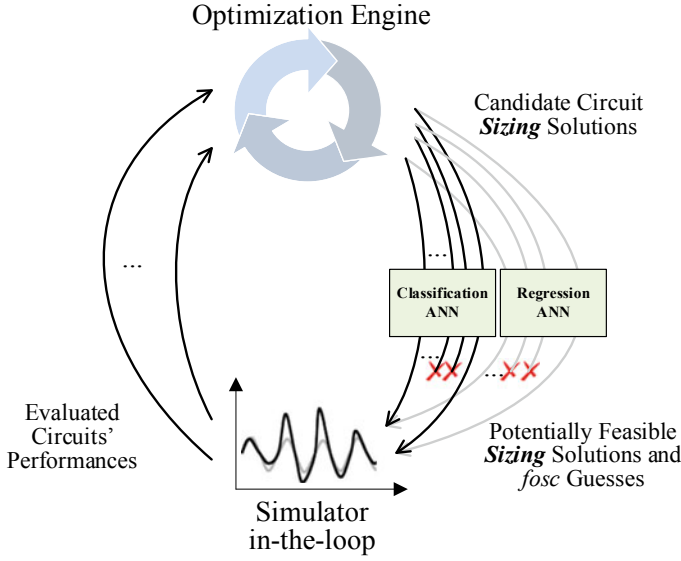


Fig. 3.1 Proposed enhanced simulation-based loop

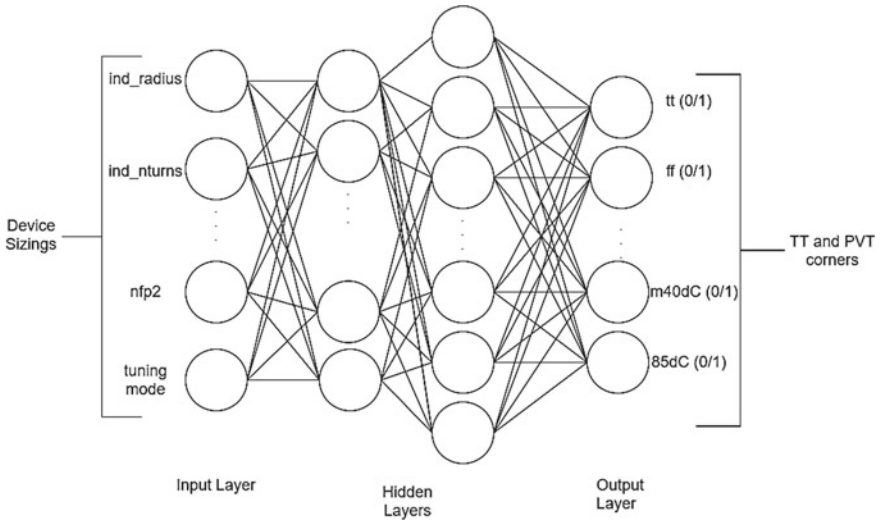


Fig. 3.2 Structure of the classification ANN

that will not be simulated. The target model will have a general behavior for the circuit analyzed, independently of the designer's constraints. Since the solution only depends on the device sizing information, the trained model can be reused on different optimizations for the same circuit.

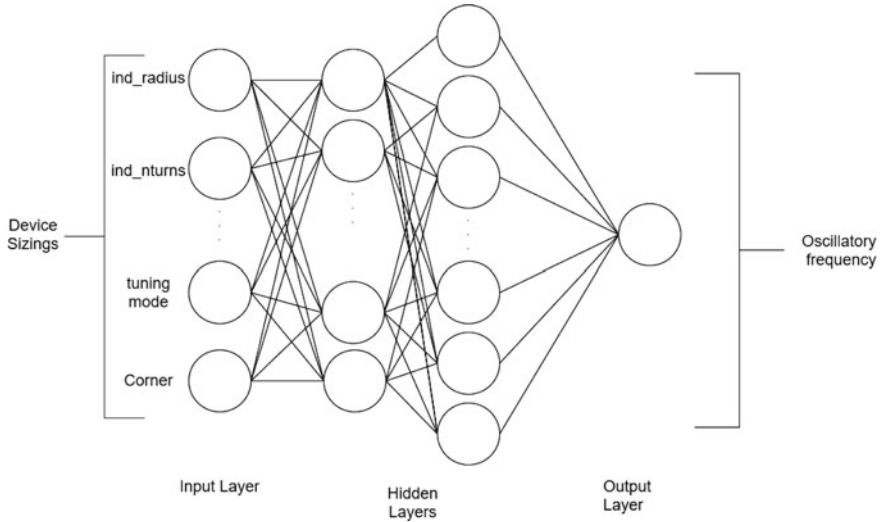


Fig. 3.3 Structure of the regression ANN

3.3 Training the Model in Isolation (Results Pre-integration)

The starting point of training the models is the preparation of the dataset. Different ANN implementations are analyzed and tested to study the impact of their hyperparameters on the convergence classification and oscillating frequency regression. Finally, the test results of some of the hyperparameters of the ANNs are described using the dataset of the complex dual-mode class C/D voltage-controlled oscillator (VCO) circuit described in Chap. 2 [23]. The class-B/C VCO from [24] is used to see how the approach achieves on another circuit using the same hyperparameters for the classification and regression ANNs. All ANNs were implemented using Python using different ML libraries such as TensorFlow [25] and Keras [26]. The code was executed on an Intel® Core™ i5-8600K 6 cores CPU 3.60 GHz with 16 GB of RAM.

3.3.1 Dataset Processing

The dataset of the complex dual-mode class C/D voltage-controlled oscillator (VCO) circuit, previously defined in Chap. 2, contains 92,115 rows of data, where for each, it is given a combination of the 28 sizing variables and its resulting 180 performance values, 90 for each tuning mode. The tuning mode is input for both the classification and regression models. Therefore, each possible combination of device sizing is replicated, and a column is added containing the tuning mode considered, resulting

in twice as many rows, i.e., 184,230 rows of data. Moreover, the 180 performance value columns were split in 90 for each of the two lines resulting from the described process. Thus resulting in a table with 184,230 lines and $28 + 1 + 90 = 119$ columns. Then, the following actions differ for each ANN:

- Classification:** For the classification network, in each row, the 10 values of performances generated for each of the 9 corners are analyzed to label the corresponding corner as “convergent” or “nonconvergent.” Each corner is examined, and if none of its performance metrics has a value of *NaN*, then the corner is labeled as “convergent.” If not, it is labeled as “nonconvergent.” Having done this for every row, the classification dataset is prepared, totaling 184,230 rows and 38 columns, 28 sizing variables and the tuning mode for the inputs, and the convergence label of the 9 corners as outputs. This process is illustrated in Fig. 3.4;
- Regression:** For the regression ANN the dataset had to go through a more thorough pre-processing. Since this ANN will only be used in convergence scenarios, only convergence scenarios should be considered in this dataset to focus the model’s training in the region in which it will operate. Moreover, each corner is considered as input. Therefore, each row of data is replicated 9 times so that for each possible combination of sizing variables, tuning mode, and corner, there is an oscillatory frequency guess, as shown in Fig. 3.5. Then, only rows (i.e., sizing, tuning mode, and corner combinations) where convergence occurs are kept. After a careful analysis, the data had to go through a final procedure to remove any possible

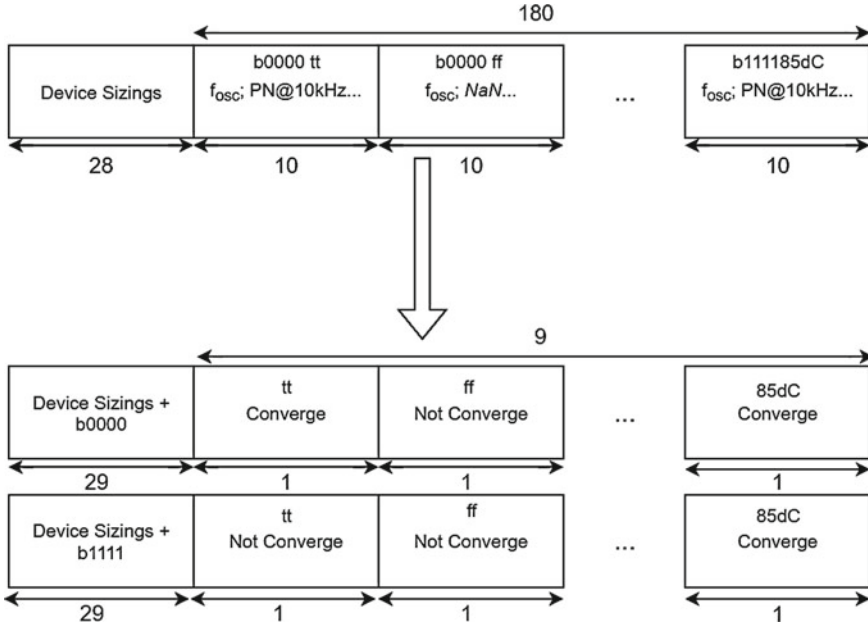


Fig. 3.4 Pre-process of classification dataset

outliers by removing the outermost 1% of the data distribution. After this step, the dataset is ready for training the ANNs. Table 3.1 showcases the data and highlights the differences observed after removing the outliers.

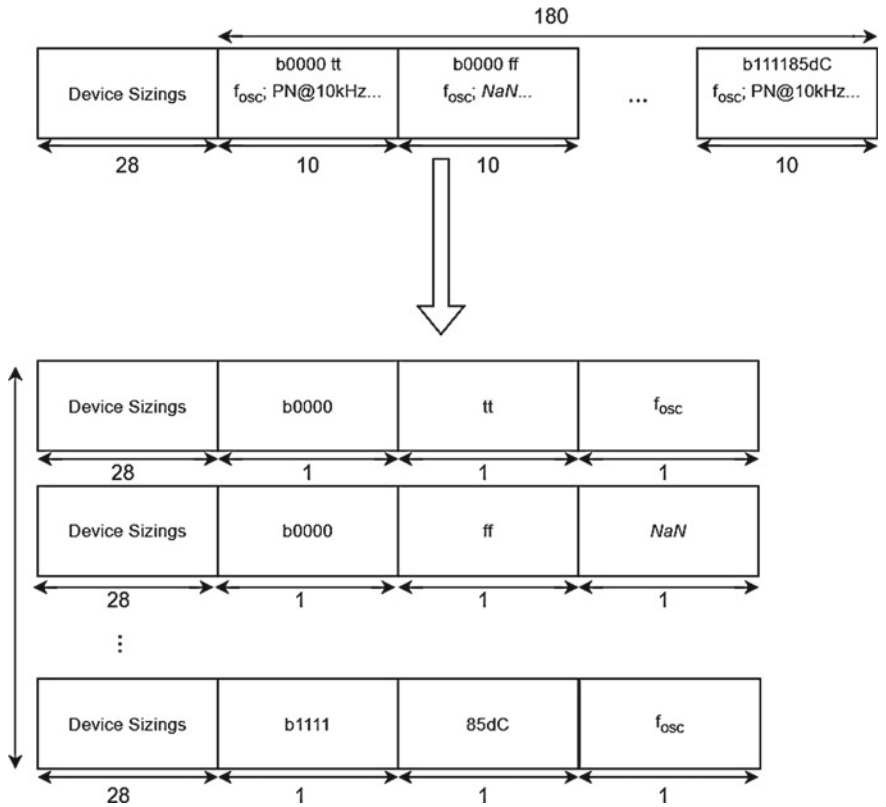


Fig. 3.5 Pre-process of regression dataset

Table 3.1 Frequency distribution with and without outliers

	Outliers	Without outliers
Max. oscillatory frequency	1.63×10^{16}	6.47×10^9
Min oscillatory frequency	-1.03×10^2	1.07×10^9
Standard deviation	1.71×10^{13}	7.93×10^8
Mean	4.16×10^{10}	4.21×10^9
Quantile 0.25	3.58×10^9	3.59×10^9
Quantile 0.5	4.14×10^9	4.14×10^9
Quantile 0.75	4.89×10^9	4.88×10^9

3.3.2 Feature Engineering

Once the data is organized, the features still need to be prepared for an adequate input dataset. The scale and distribution of the data are significant to consider towards a stable numerical process. Some features may have different scales, while others may not even have units. If not adequately scaled, these variations may lead to either very large or minimal gradients, introducing a bias towards larger-scale features. In this work, we considered min–max scaling and standardization is considered.

Min–max scaling scales a distribution to a range, typically between 0 and 1. This is obtained through (3.1), where x is the unscaled feature distribution and x' is the scaled distribution:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (3.1)$$

In standardization, a feature is modeled as a normal distribution centered in 0 with a unitary standard deviation. This is done through:

$$x' = \frac{x - \mu}{\sigma} \quad (3.2)$$

where μ and σ are the mean and standard deviation of the unscaled feature distribution, respectively.

3.3.3 Convergence Classifier and Its Hyperparameters

Having the dataset prepared, it can be used to train the classifier. Firstly, the data is divided into train and test data with a ratio of 80% and 20%, respectively, and where the data is first randomly shuffled before splitting. Regarding the classification, the metrics used to evaluate the model were:

- **Loss**, computed by the binary cross entropy (3.3), in which the predictions made are compared to the collected labels;

$$L = -\frac{1}{n} \sum_{i=1}^n y_i * \log \hat{y}_i + (1 - y_i) * \log(1 - \hat{y}_i) \quad (3.3)$$

- **Binary accuracy**, which computes the rate of correct predictions made by the model, i.e., how often the model's predictions align with the labels and is given by (3.4), where *True Positives (TP)* is the number of points correctly predicted as convergent, *True Negatives (TN)* the number of points correctly

predicted as nonconvergent, *False Positives (FP)* are nonconvergent points incorrectly predicted as convergent and *False Negatives (FN)* are convergent points incorrectly predicted as nonconvergent;

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.4)$$

- **Recall** evaluates how many of the dataset's convergence scenarios were correctly predicted by the model, as in (3.5).

$$Recall = \frac{TP}{TP + FN} \quad (3.5)$$

Besides training the model's parameters (weights and biases), the model's hyperparameters substantially impact the training time and final performance. Thus, these hyperparameters should be carefully selected. A simplified empirical search of the hyperparameter space is conducted toward this goal, where, even though these hyperparameters interact, a simplified approach is taken, and each hyperparameter is optimized individually, with all others fixed. Although no established order exists to conduct this study, an effort was made to tune the hyperparameters in order of estimated impact on model performance (most impactful first).

Training is done with the Adam optimizer, which has three hyperparameters, the default values for β_1 , β_2 and ϵ are robust to a variety of settings. Thus, no optimization of these hyperparameters is conducted, and the default values corresponding to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and ϵ to a small value, such as 10^{-8} , were used. We start with the ReLU as the activation function for the hidden layers and sigmoid for the output layer, the simple architecture of an ANN with two hidden layers with 200 neurons per layer, and MinMax scaling with a range going from 0 to 1 as the normalization scheme. A dropout of 20% is used to prevent overfitting and add robustness [27].

As a starting point, the learning rate is the first hyperparameter to tune since its impact on model performance is expected to be the highest of all [28]. The learning rate dictates how fast the model responds to the estimated error in each weight update. Setting this value is challenging. The process of choosing the most appropriate learning rate is not very well defined but rather empirical, where the approach starts with a high learning rate and steadily decreases it until no improvement is observed. The results in Table 3.2 show that the value of 0.003 presents the best overall results, with minimum training and test loss, maximum training and test accuracy, and third-best train and test recall.

An activation function defines the expressiveness of each neuron. In regression, when predicting the circuit's oscillating frequency, where the output is a real value, the output neuron is taken directly from its linear activation. However, for the output of the classification and the hidden layers of both the ANNs, it is necessary to have non-linear activation functions to provide expressiveness to the network. Regarding the classification model, the sigmoid function, shown in Fig. 3.6, is a prominent function used in classification networks' output layers, having the ability to map

Table 3.2 Learning rate evolution with 32 batch size, ReLU, and sigmoid activation functions, two hidden layers and 200 neurons per layer, and MinMax of 0–1

	Training loss	Test loss	Training binary accuracy	Test binary accuracy	Training recall (%)	Test recall (%)
0.009	0.1022	0.1117	0.9654	0.9628	99.36	99.25
0.007	0.1051	0.1140	0.9636	0.9604	98.18	98.03
0.005	0.0935	0.1049	0.9686	0.9652	99.41	99.25
0.003	0.0862	0.0990	0.9726	0.9678	99.13	98.87
0.0009	0.0881	0.0992	0.9716	0.9675	98.89	98.68

the input values of a neuron to a value in the 0–1 range. This function contains the neuron’s output in the feasible range of binary classification tasks, making it helpful for this class of problems.

As discussed in [29], other good candidate activation functions generally outperform the sigmoid function when addressing DNNs. The Rectified Linear Unit (ReLU) is a popular activation function, mathematically defined by $f(z) = \max(0, z)$. One issue that arises from using the ReLU activation function is denominated the dying ReLU function, in which the neuron’s parameters stop updating due to the null gradient in the $z < 0$ region, thus becoming stuck in that region. Variants of the ReLU function, such as the leaky ReLU, have been proposed to soften the hard

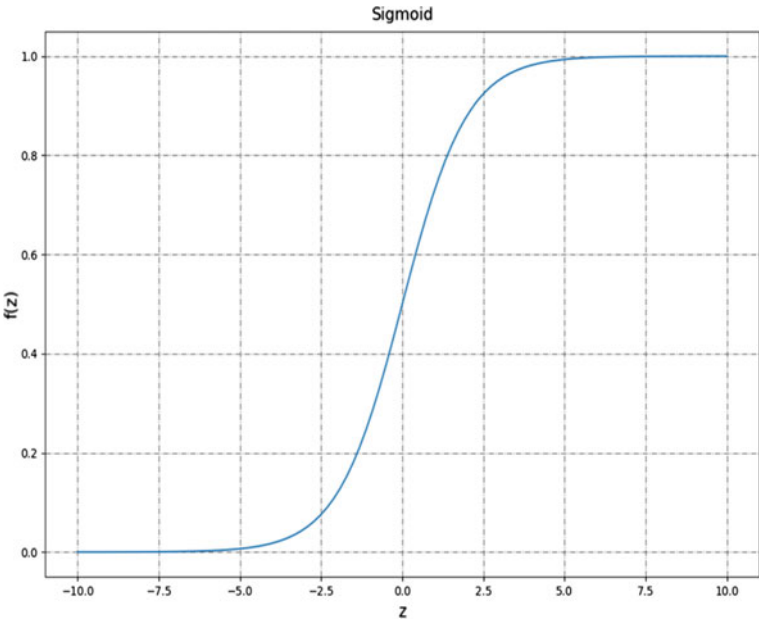


Fig. 3.6 Sigmoid function

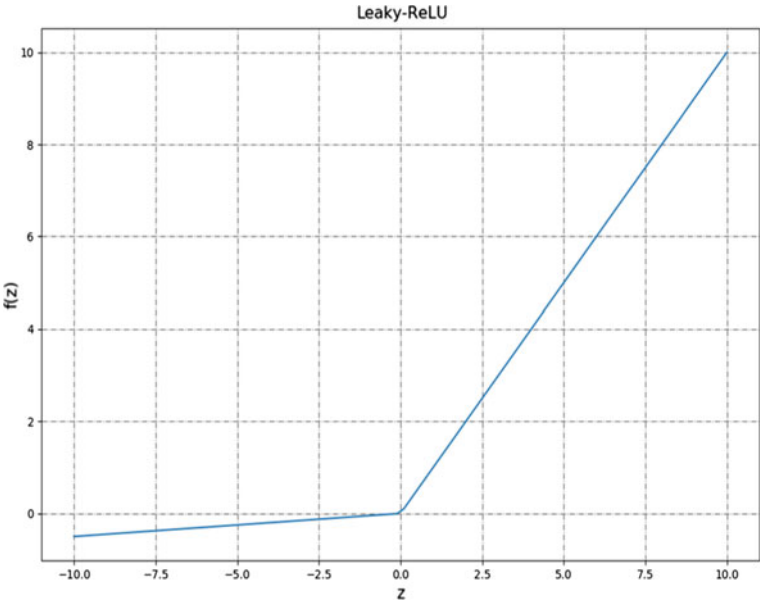


Fig. 3.7 Leaky ReLU function

cutoff the ReLU function presents in the $z < 0$ region. The leaky ReLU is illustrated in Fig. 3.7.

Regarding the activation function used in the model’s hidden layers, Table 3.3 shows the results obtained for each tested activation function. The results show that the *sigmoid* function achieves the best test loss and accuracy while achieving a comparable test recall score, thus presenting the best overall performance.

The size (number of layers and number of neurons in each layer) determines the expressiveness of the ANN. Whereas a small model is easy to train and is unlikely to overfit, it will not be able to represent very complex notions. Conversely, large models

Table 3.3 Hidden layers activation functions evolution with 0.003 learning rate, 256 batch size, two hidden layers and 200 neurons per layer and MinMax of 0–1

	Training loss	Test loss	Training binary accuracy	Test binary accuracy	Training recall (%)	Test recall (%)
ReLU	0.0925	0.1052	0.9686	0.9642	98.50	98.26
Leaky ReLU	0.0953	0.1047	0.9674	0.9639	99.40	99.31
Elu	0.0983	0.1022	0.9669	0.9656	99.23	99.15
Tanh	0.1008	0.1100	0.9657	0.9617	99.25	99.12
Sigmoid	0.0887	0.1010	0.9711	0.9667	99.10	98.90

Table 3.4 Number of layers and neurons evolution with 0.003 learning rate, 256 batch size, sigmoid activation functions, and MinMax of 0–1

	Training loss	Test loss	Training binary accuracy	Test binary accuracy	Training recall (%)	Test recall (%)
200/200	0.0910	0.1028	0.9693	0.9648	99.29	99.16
200/200/150	0.0876	0.0987	0.9719	0.9677	99.14	98.98
200/200/175	0.0879	0.0983	0.9716	0.9673	98.87	98.65
200/200/200	0.0873	0.0977	0.9715	0.9678	99.01	98.83
200/200/225	0.0864	0.0968	0.9721	0.9682	98.76	98.54
200/200/250	0.0851	0.0941	0.9730	0.9694	99.11	98.95
200/200/275	0.0851	0.0955	0.9730	0.9687	99.01	98.80
200/200/250/200	0.0900	0.0982	0.9709	0.9679	98.90	98.72

can represent higher dimension problems but are more likely to overfit. Moreover, large models also increase the time and computational effort for the training stage, as the output of each consequent layer is computed sequentially (not parallelized). As there is no formula to choose the optimal number of hidden layers, one single hidden layer will be implemented, and their number will be incremented until an increase in the test error is observed, after which point, the model is assumed to overfit further.

The number of neurons to tune only applies to the neurons in the hidden layers since the ones in the input correspond to the number of sizing parameters plus the tuning mode and the output to the number of corners. Table 3.4 shows that when using three layers with 200, 200, and 250 neurons, respectively, the best values of loss and accuracy are achieved, only falling behind in recall, thus presenting the best overall scores.

Finally, different ranges of MinMax scaling and its use were tuned and compared, arriving at the results shown in Table 3.5. The values of loss and accuracy showed to be the best when using MinMax with a range going from 0 to 1, making the slight variations of recall values when using different ranges not worth the tradeoff. Thus, MinMax normalization to a range between 0 and 1 is used.

3.3.4 Regressor and Its Hyperparameters

The hyperparameter tuning process for the frequency guess regression model was conducted similarly to the ones of the classifier. However, since the task differs, the metrics used to evaluate the hyperparameters and the possible hyperparameter values explored differ.

- **Loss**, when considering regression, where the frequency value is predicted, mean squared error (MSE) is the most appropriate function to use. Equation (3.6) shows

Table 3.5 Normalization evolution with 0.003 learning rate, 256 batch size, sigmoid activation functions, and three hidden layers with 200, 200 and 250 neurons per layer, respectively

	Training loss	Test loss	Training binary accuracy	Test binary accuracy	Training recall (%)	Test recall (%)
No MinMax	0.1785	0.1927	0.9372	0.9319	98.67	98.38
MinMax (0,1)	0.0852	0.0963	0.9727	0.9686	99.13	98.87
MinMax (0.5,1.5)	0.0993	0.1038	0.9679	0.9666	99.06	99.02
MinMax (1,2)	0.1080	0.1115	0.9629	0.9613	99.29	99.25

the loss used: the average squared difference between the simulated frequency, f_i , and the one predicted, $\hat{f}_i$.

$$L = MSE = \frac{1}{n} \sum_{i=1}^n (f_i - \hat{f}_i)^2 \quad (3.6)$$

- **Mean Absolute Error (MAE)**, which gives the mean of the absolute difference between the predictions made and the labels;

$$MAE = \frac{1}{n} \sum_{i=1}^n |f_i - \hat{f}_i| \quad (3.7)$$

- **Mean Absolute Percentage Error (MAPE)**, a measure of how accurate the model is considering the scale of the data.

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{f_i - \hat{f}_i}{f_i} \right| \quad (3.8)$$

As a starting point, the same logic was used as before. Therefore, the hyperparameters were set to leaky ReLU as the activation functions, MinMax scaling with a range going from 0 to 1, and two hidden layers with 200 neurons per layer. Following the same logic, the learning rate started at a relatively high value and gradually decreased until no further improvement was observed. Analyzing Table 3.6, the best values of MSE and MAE come from using a value of 0.0008, and so this value was chosen.

Analyzing the results, it is possible to observe that the values of the MAPE were too high. The normalization scheme could drastically influence this value, thus, this parameter is the next to be optimized, changing the order relative to the classification model. As Table 3.7 shows, varying the minmax range greatly affects the MAPE value, the best results of MSE and MAE come from using MinMax with a range of

Table 3.6 Learning rate evolution with leaky ReLU activation functions, two hidden layers and 200 neurons per layer, and MinMax of 0–1

	Training MSE	Validation MSE	Training MAE	Validation MAE	Training MAPE	Validation MAPE
0.004	0.0021	0.0021	0.0292	0.0293	122.10	9.2120
0.0008	0.0009	0.0009	0.0170	0.0170	152.46	7.6622
0.0007	0.0011	0.0011	0.0212	0.0212	180.91	6.7929
0.0006	0.0015	0.0015	0.0268	0.0267	288.66	13.586
0.0005	0.0009	0.0008	0.0173	0.0173	72.251	361.88

0–1. However, the MAPE values are too high, so, MinMax was used with a range of 1–2, since, overall, it has the best results, achieving the lowest values of MAPE.

Regarding the activation function used in the hidden layers (the activation function in the output layer is the linear activation function), Table 3.8 shows that the Leaky ReLU results in the best overall performance.

Finally, the number of layers and neurons per layer were tuned, having concluded that two layers with 200 and 150 neurons per layer, respectively, was the best option, as the lowest values of MSE, MAE, and MAPE are obtained, as shown in Table 3.9.

Table 3.7 Normalization evolution with 0.0008 learning rate, leaky ReLU activation functions, two hidden layers and 200 neurons per layer

	Training MSE	Test MSE	Training MAE	Test MAE	Training MAPE	Test MAPE
No MinMax	0.1896	0.1889	0.3106	0.3108	177.45	203.70
MinMax (0,1)	0.0008	0.0007	0.0172	0.0172	99.090	224.68
MinMax (0.5,1.5)	0.0011	0.0011	0.0197	0.0196	2.0301	2.0290
MinMax (1,2)	0.0011	0.0011	0.0199	0.0199	1.3332	1.3327

Table 3.8 Activation functions evolution with 0.0008 learning rate, 256 batch size, two hidden layers and 200 neurons per layer and MinMax of 1–2

	Training MSE	Test MSE	Training MAE	Test MAE	Training MAPE	Test MAPE
ReLU	0.0012	0.0012	0.0224	0.0225	1.4365	1.4402
Leaky ReLU	0.0010	0.0010	0.0190	0.0191	1.2736	1.2816
Elu	0.0011	0.0011	0.0217	0.0218	1.4686	1.4709
Tanh	0.0012	0.0012	0.0195	0.0201	1.3290	1.3891
Sigmoid	0.0011	0.0011	0.0231	0.0231	1.5500	1.5499

Table 3.9 Number of layers and neurons evolution with 0.0008 learning rate, 256 batch size, leaky ReLU activation functions, and MinMax of 1–2

	Training MSE	Test MSE	Training MAE	Test MAE	Training MAPE	Test MAPE
200/125	0.0005	0.0006	0.0121	0.0121	0.7928	0.7964
200/150	0.0005	0.0005	0.0100	0.0101	0.6486	0.6525
200/175	0.0005	0.0005	0.0117	0.0117	0.7511	0.7532
200/200	0.0007	0.0007	0.0139	0.0139	0.8601	0.8640
200/150/75	0.0007	0.0007	0.0149	0.0150	0.9514	0.9587
200/150/100	0.0006	0.0006	0.0124	0.0123	0.7749	0.7708
200/150/125	0.0007	0.0007	0.0164	0.0164	1.0659	1.0679
200/150/150	0.0006	0.0006	0.0129	0.0129	0.8454	0.8489

3.3.5 Final Model Details

The hyperparameter tuning process led to a classification model with the hyperparameters described in Table 3.10 and a loss and accuracy curve of both training and test, as shown in Figs. 3.8 and 3.9. An analysis of the loss curve shows a monotonic decreasing trend as epochs increment, thus suggesting a stable convergence process. At the last few epochs, the test loss starts exhibiting increasing instability and growth, thus suggesting that further training could lead to overfitting. As for the accuracy curves: consistent improvement and increasing test instability in the last few epochs. The final accuracy values indicate that the model has a probability of less than 5% to classify the convergence of a corner incorrectly.

Furthermore, two more metrics emphasized the model’s validity: precision and F1 score. Precision shows the portion of correct identifications, while F1 is an overall measure of a model’s accuracy resulting from the harmonic mean of the precision

Table 3.10 Summary of the classification model

Hyperparameter	Value
Input layer	1 layer (29 neurons)
Hidden layers	3 layers (200,200,250 neurons)
Output layer	1 layer (9 neurons)
Activation functions	Sigmoid
Optimizer	Adam
Regularizer	Dropout (drop rate = 20%)
Loss function	Binary crossentropy
Learning rate	0.003
Epochs	200
Batch size	256
Normalization	Min Max (0,1)

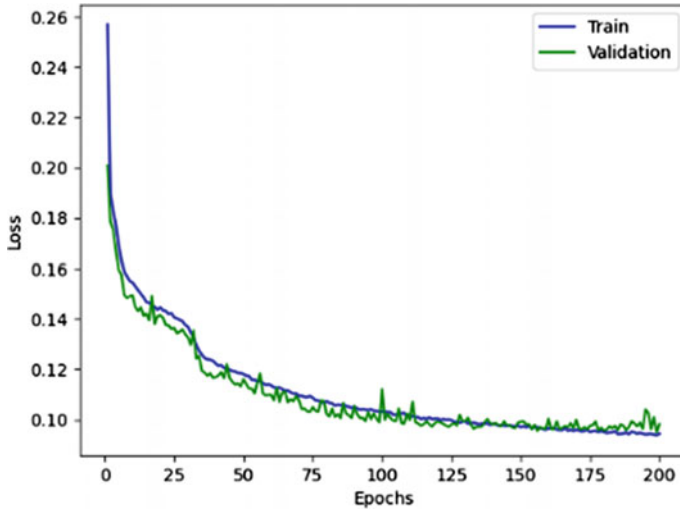


Fig. 3.8 Loss

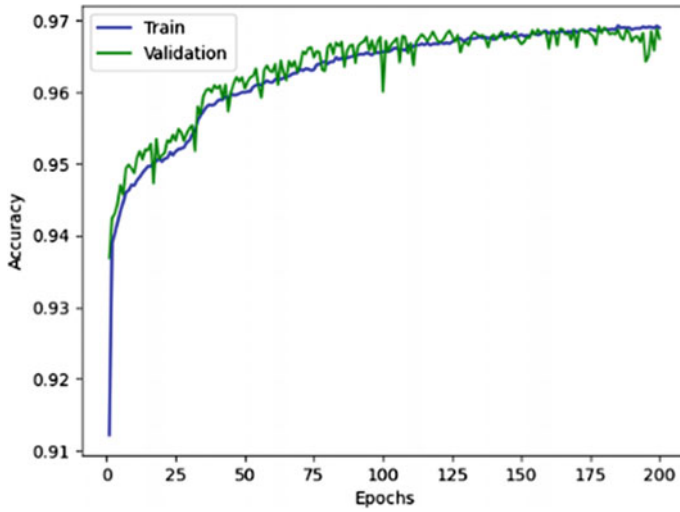


Fig. 3.9 Accuracy

and recall metrics. The formulas of precision and F1 score are shown in (3.9) and (3.10), respectively, and their results on Table 3.11.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.9)$$

Table 3.11 Train and test precision and F1 score results

	Train	Test
Precision	0.979	0.977
F1 score	0.985	0.983

Table 3.12 Summary of the regression model

Hyperparameter	Value
Input layer	1 layer (30 neurons)
Hidden layers	2 layers (200,150 neurons)
Output layer	1 layer (1 neuron)
Activation functions	Leaky ReLU and linear
Optimizer	Adam
Regularizer	Dropout (drop rate = 20%)
Loss function	MSE
Learning rate	0.0008
Epochs	100
Batch size	256
Normalization	Min Max (1,2)

$$F1 \text{ score} = \frac{2 * TP}{2 * TP + FP + FN} \quad (3.10)$$

The regressor was implemented using the hyperparameters shown in Table 3.12. The curves for all metrics for both the train and test sets are shown in Figs. 3.10, 3.11, and 3.12. Analyzing the graphs, the model learns to predict the oscillatory frequency inferred by the decreasing metric values throughout training.

3.3.6 Discussion

For the class-C/D VCO, the results shown are promising, showing good performance in the samples outside the training set, i.e., unseen data. Additionally, by analyzing the histograms in Figs. 3.13 and 3.14, it is possible to see that the frequency distribution of the test label set and the test predictions set do not differ significantly. In addition, a detailed analysis was conducted, and the results from the worst, medium, and best MAPE results, as well as the predicted values and the real ones, are presented in Table 3.13, showing that, even though the worst results are still high, the overall results show that the ANN can predict with a firm assurance the oscillatory frequency value. Therefore, it is reasonable to assume that the model's training was successful.

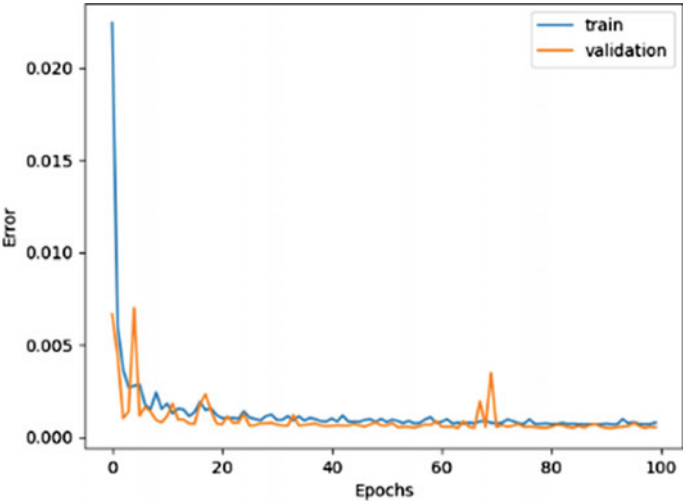


Fig. 3.10 Training and test MSE

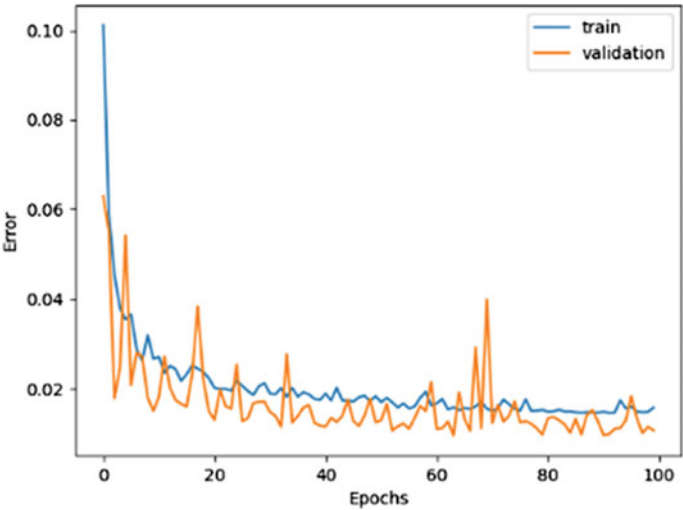


Fig. 3.11 Training and test MAE

3.4 In-the-Loop Integration

Next, the convergence classifier and frequency guess regressor are included in AIDA’s simulation-based multi-objective multi-constraint sizing optimization loop []. The comparison and analysis between the optimization results obtained with and without using ANNs are performed and discussed.

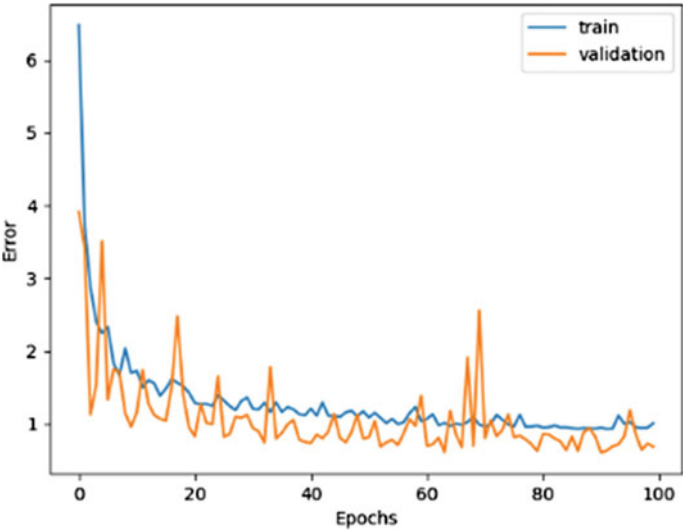
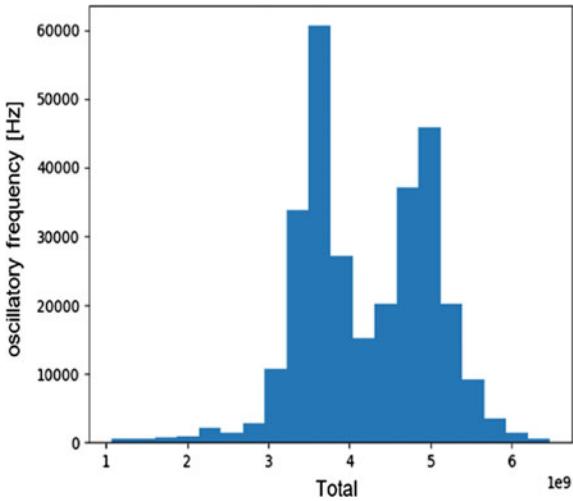


Fig. 3.12 Training and test MAPE

Fig. 3.13 Real frequency values



The classification ANN discards a point (i.e., a candidate sizing solution) based on if the percentage of corners predicted to converge lower than some threshold, then the point is discarded. Regarding the predicted oscillatory frequency, it was possible to conclude that when predicted values were negative, even if the classification predicted many corners to converge, the simulator would not be able to obtain values for that point. Therefore, in case of negative values given by the regression ANN, the point in question will not be sent through the simulator, as illustrated in Fig. 3.15. Different

Fig. 3.14 Predicted frequency values

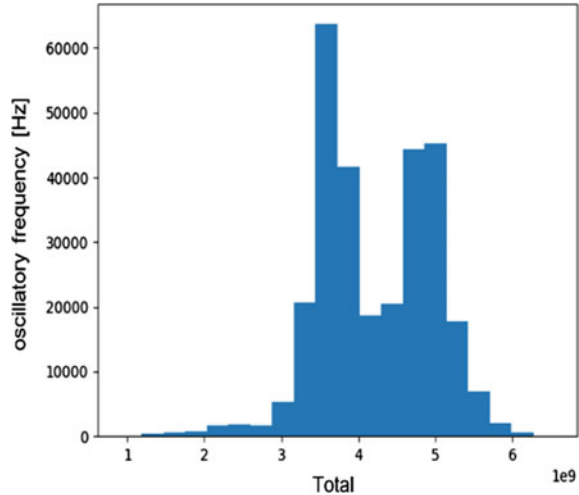


Table 3.13 Test MAPE results and difference between predicted frequency values and real ones

	Test MAPE	Test prediction normalized	Test prediction de-normalized (GHz)	Real value normalized	Real value de-normalized (GHz)
Worst values	9.22e+01	1.93	6.08	1.00	1.09
	8.68e+01	1.91	5.96	1.02	1.17
	8.60e+01	1.93	6.09	1.04	1.27
	8.41e+01	1.87	5.77	1.02	1.16
	8.22e+01	1.84	5.62	1.01	1.13
Medium values	5.97e−01	1.47	3.62	1.46	3.57
	5.97e−01	1.73	5.00	1.72	4.95
	5.97e−01	1.77	5.21	1.78	5.27
	5.97e−01	1.67	4.69	1.66	4.64
	5.97e−01	1.49	3.73	1.48	3.68
Best values	2.08e−05	1.57	4.17	1.57	4.17
	2.00e−05	1.70	4.82	1.70	4.82
	1.69e−05	1.46	3.54	1.46	3.54
	1.65e−05	1.60	4.32	1.60	4.32
	1.64e−05	1.93	5.01	1.00	1.09

thresholds to determine if the points will be sent to the simulator are considered, and the results are analyzed. Not only did the threshold come from the usage of the classification ANN but also the regression ANN. The functioning of this threshold is illustrated in Fig. 3.16.

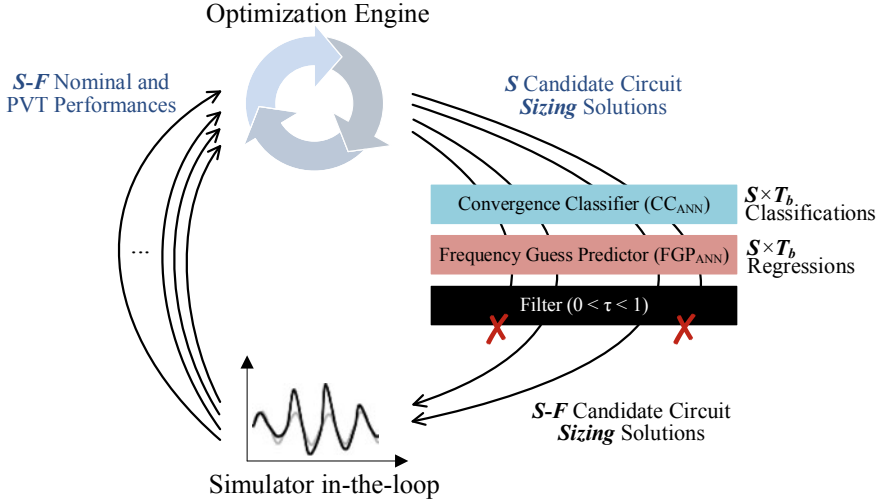


Fig. 3.15 Introduction of the threshold in a simulation-based sizing flow

An optimization without the filters is used as a baseline to assess the advantages of having these filters. As a starting point, the points that form the original Pareto optimal front (POF) obtained were passed through the ANNs. The outcome of the ANNs showed that all the points in the POF were also validated with the ANNs. Therefore, they would be presented to the simulator in case they appeared during the optimization.

3.4.1 Class C/D VCO for 3.5-to-4.8 GHz @ 50% Threshold

To start, the directive to allow the points to be simulated was a naive approach starting with a 50% threshold, meaning that if the classification ANN predicts, for a given point, that more than half the PVT corners converge, then that point is simulated, otherwise, it is discarded. The exact configuration is used for optimizations with and without the filters, using a population of 256 elements and 150 generations.

During the optimization, the points simulated and discarded were registered, resulting in the ratio of points simulated illustrated in Fig. 3.17. It shows that, from the points supposed to be simulated, 18.65% were discarded. More, it is important to note that not all simulations take the same time. The simulation can be 100 times longer when the simulator encounters convergence issues. Since the filtered points are more likely to be the source of convergence issues, the impact on the execution time will be superior. Since the optimization without the filters lasted 25 days, more than 5 days were economized with the filter.

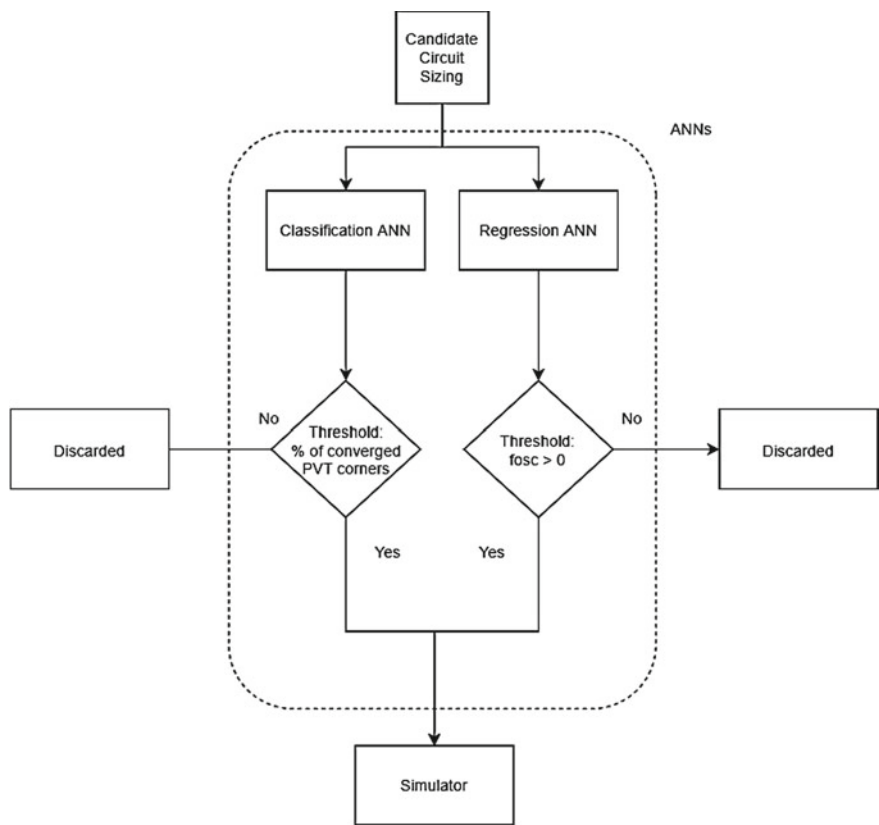
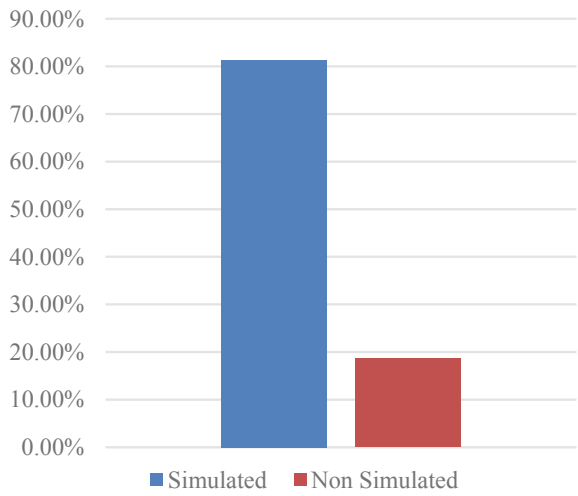


Fig. 3.16 Threshold function in detail

Fig. 3.17 Ratio of points discarded using a value of 50% threshold



With these results, a deeper analysis was conducted to test the accuracy of the values of oscillatory frequency predicted by the regression ANN. The evolution of the predicted oscillatory frequency along the generations was registered. The difference between those and those given by the simulator was outputted using the MAPE formula. For the purpose of this study, 5 random points were taken for every generation studied.

The results for the first generation are presented in Fig. 3.18, where the MAPE presents some high values, reaching even values of 41.55% discrepancy for one point. This means that the predicted values were considerably far from the ones the simulator gave. As the optimization evolves, the values given by the regression ANN start to get closer, and the MAPE values start to decrease, never reaching values higher than 14%, as observed in Fig. 3.19, where the results were made using the values obtained from the 75th generation.

Finally, the results of the last generation, i.e., the 5 random points from the 150th generation, show auspicious figures, achieving MAPE values lower than 6%, as observed in Fig. 3.20. These results suggest that the evolutionary process quickly converges to the general region of the optimal value, thus, the generated dataset through an evolutionary process contains many data points in this region, and so, the model is better fine-tuned for the later stages of the optimization process.

By inspecting the results, it is possible to conclude that the error between the value predicted by the regression ANN is high as the optimization begins but decreases over generations, culminating in values relatively close to the ones given by the simulator. It is also deduced that, along the graphs, the corner SS has the highest

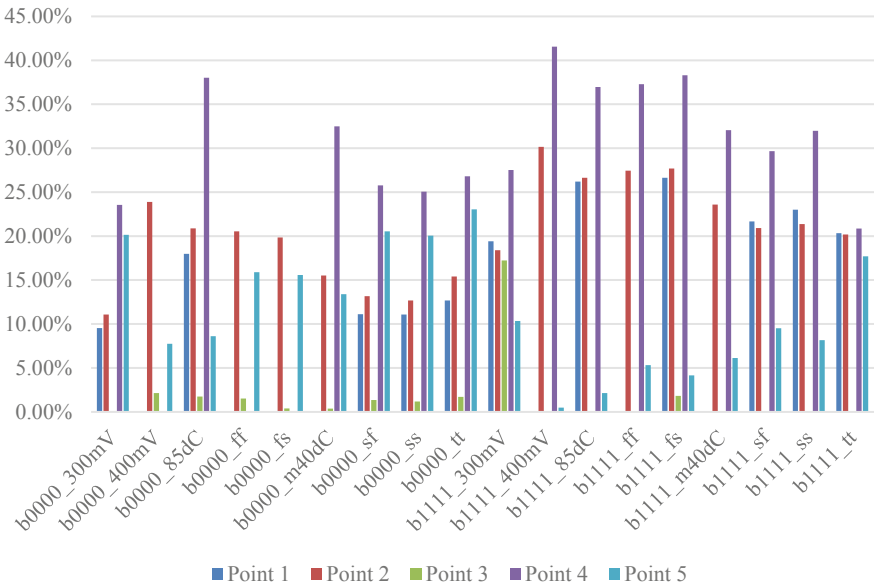


Fig. 3.18 MAPE values from 5 random points of the first generation using a 50% threshold

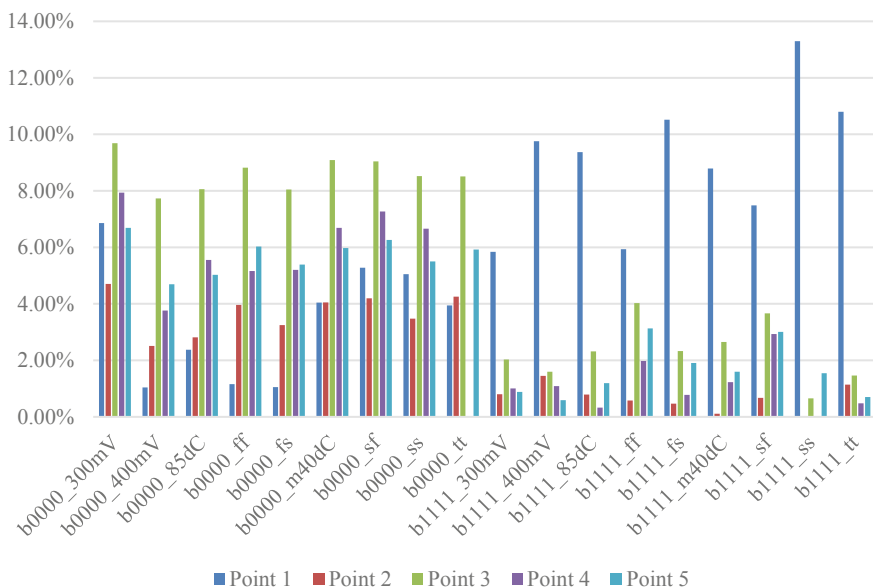


Fig. 3.19 MAPE values from 5 random points of the 75th generation using a 50% threshold

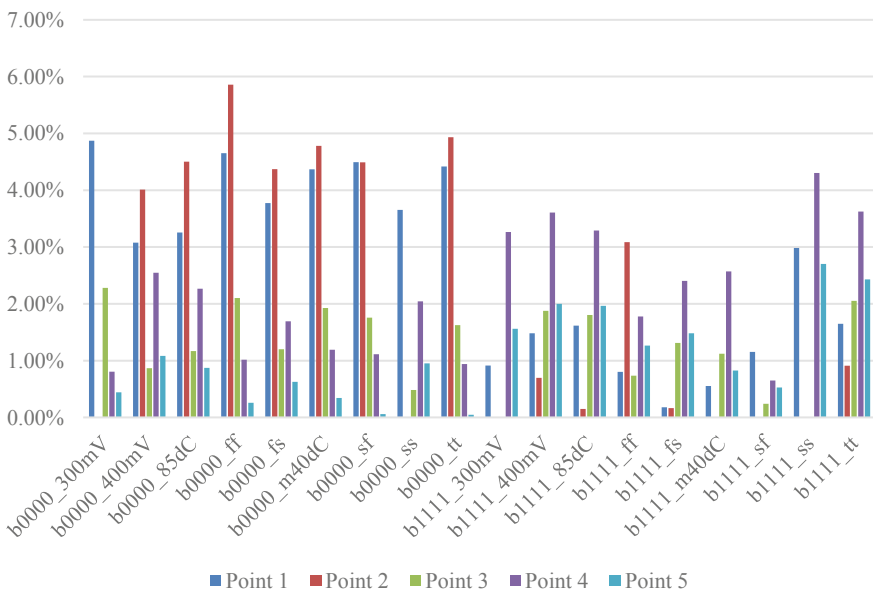
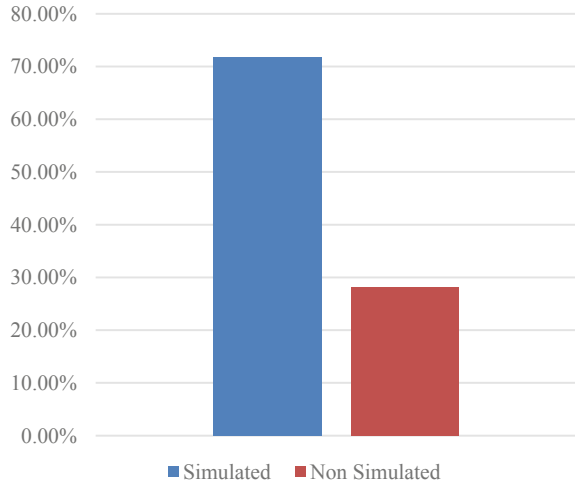


Fig. 3.20 MAPE from 5 random points of the 150th generation using a 50% threshold

Fig. 3.21 Ratio of points discarded without the use of the regression ANN



error, suggesting that this might be the most challenging condition to evaluate the circuit.

3.4.1.1 Impact of the Frequency Guess

To evaluate the impact of the frequency guess predictor, another optimization with the same characteristics was conducted with the model removed, with a population of 256 elements optimized for 150 generations, leading to a discard of almost 30% of the total points that would be simulated, which was higher than the previous results, as Fig. 3.21 shows. This increase in points discarded can be explained since, without the frequency guess predictor, the simulator could not converge in some designs, hindering the optimization and leading to more solutions being discarded by the classification ANN.

The optimal points for each optimization, with and without the regression ANN, were retrieved, and a POF was obtained. These two POFs were then compared against the POF obtained without using the ANNs, considered the reference, as depicted in Fig. 3.22. As the results show, the use of the regression ANN proves to be valuable since even though its corresponding POF shows that the solutions obtained have worse results in terms of power, never reaching values below $1.30\text{E}-03$, the phase noise results are considerably better. Some solutions have worse values of power. However, some achieve values lower than $9.00\text{E}-04$ and better results in phase noise, reaching values of -136.00 dBc/Hz. Therefore, using the regression ANN proves to be essential, as it helps the simulator converge thanks to the predicted value of the oscillatory frequency.

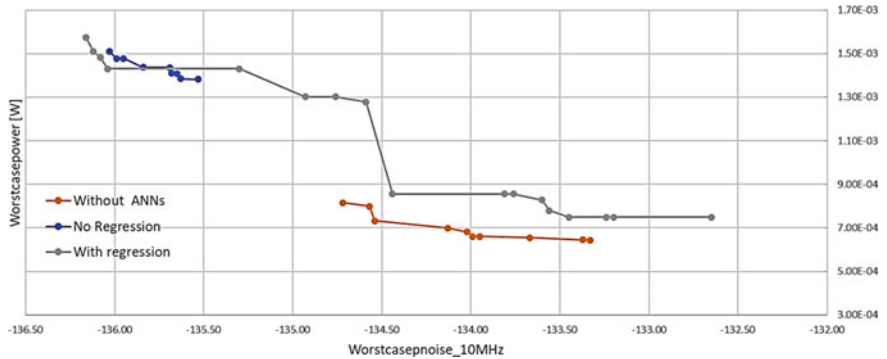


Fig. 3.22 Comparison between the obtained POFs

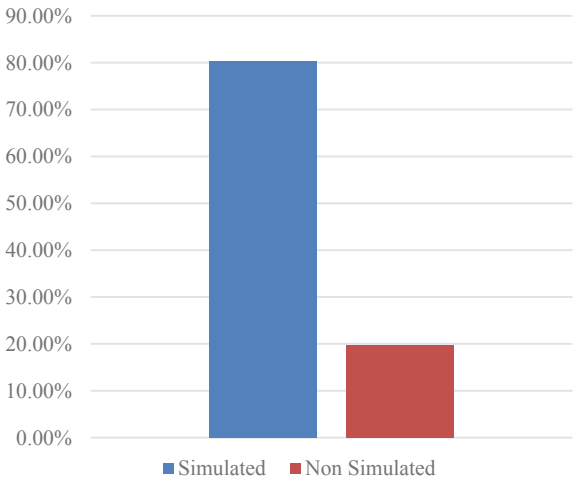
3.4.2 Class C/D VCO for 3.5-to-4.8 GHz @ 75% Threshold

Next, the threshold was set to 75%, being this value is more rigid, so the number of points discarded is expected to be higher. The same configuration for the optimization was used, so the simulated points were computed, reaching the ratio showcased in Fig. 3.23. As expected, the rate of non-simulated points was higher, reaching a value of 19.65% of points discarded from the total points that would be fed to the simulator.

The MAPE values along the optimization cycle were measured, are presented in Figs. 3.24, 3.25, and 3.26 the results for the first, 75th and 150th generation, respectively.

As expected, the MAPE values kept decreasing as the generations increased, starting at values of 37% up to the point of reaching values under 12%, even though the results are not as spread as the ones from Sect. 3.4.1. The resulting POF was

Fig. 3.23 Ratio of points discarded using a value of 75% threshold



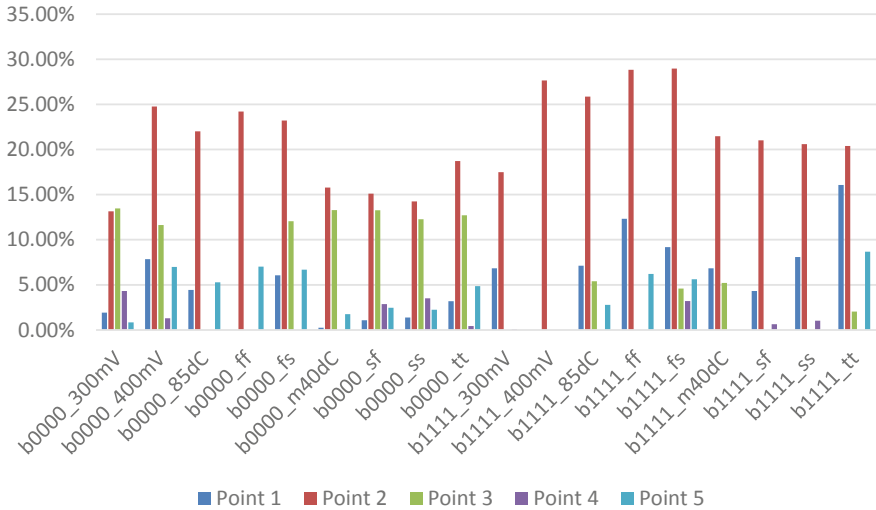


Fig. 3.24 MAPE values from 5 random points of the first generation using a 75% threshold

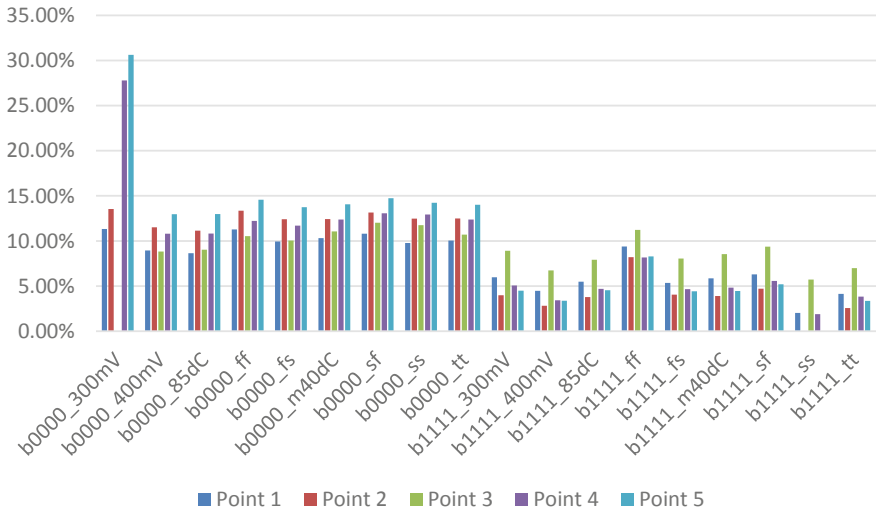


Fig. 3.25 MAPE values from 5 random points of the 75th generation using a 75% threshold

again compared with the reference POF as illustrated in Fig. 3.27, showing that the points obtained were similar to the reference one, reaching power and phase noise values better than the unaltered optimization cycle. In terms of power, all the values are lower than $7.91\text{E}-04$ and lower than -134.19 in phase noise.

To sum up, the results show that the ANNs are accomplishing what is expected, discarding unwanted solutions, being each ANN indispensable. They show apt to

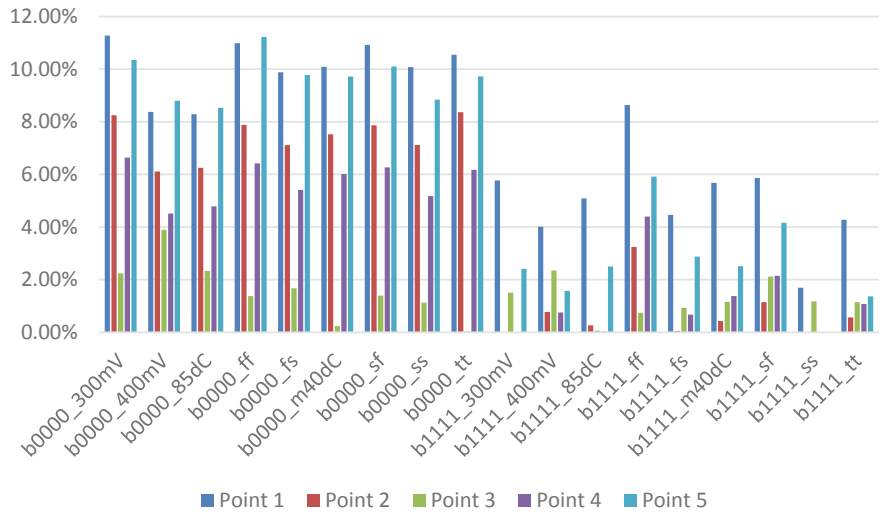


Fig. 3.26 MAPE values from 5 random points of the 150th generation using a 75% threshold

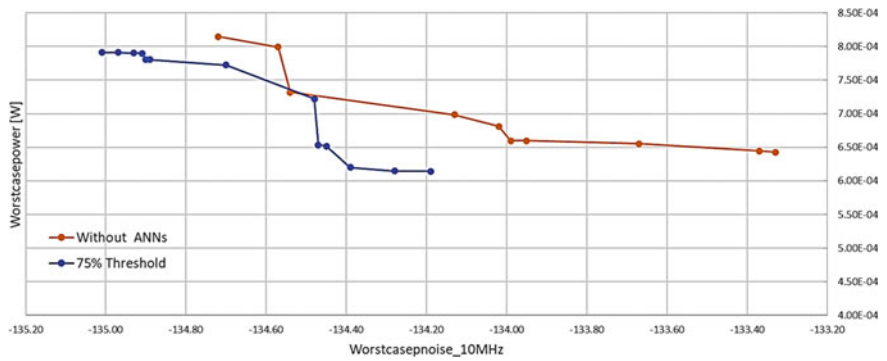


Fig. 3.27 POF obtained with a 75% threshold

perform optimizations in less time, without compromising the results. Next the threshold is set up to an even higher value as the value is set to 90 and 100%, in order to better comprehend the impact of this parameter in the optimization cycle.

3.4.3 Class C/D VCO for 3.5-to-4.8 GHz @ 90% and 100% Thresholds

After setting the threshold to the higher values, the same logic and optimization configuration was used as before, so the number of points discarded from the total

number was again registered. The results are shown in Fig. 3.28, where it can be seen that the number of filtered samples was lower than the ones obtained from the use of a 75% threshold.

From the inspection of the POFs obtained in Figs. 3.29 and 3.30, it is possible to conclude that the results were worse, as the POFs obtained are further away from the reference POF. With a 90% value threshold, some points have good phase noise values. However, the power values are too high, whereas with a 100% threshold, both the power values and phase noise are considerably worse.

Therefore, not only were the number of points discarded lower, but also the values of the optimization worse, which led to the conclusion that such high threshold values come at a performance cost, as promising solutions are discarded with no chance to

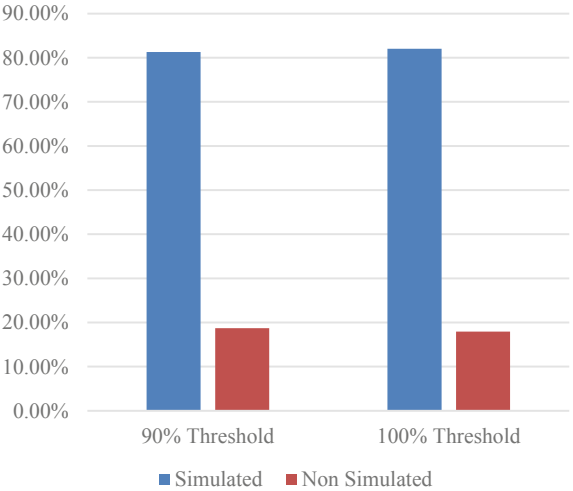


Fig. 3.28 Ratio of points discarded with 90 and 100% threshold

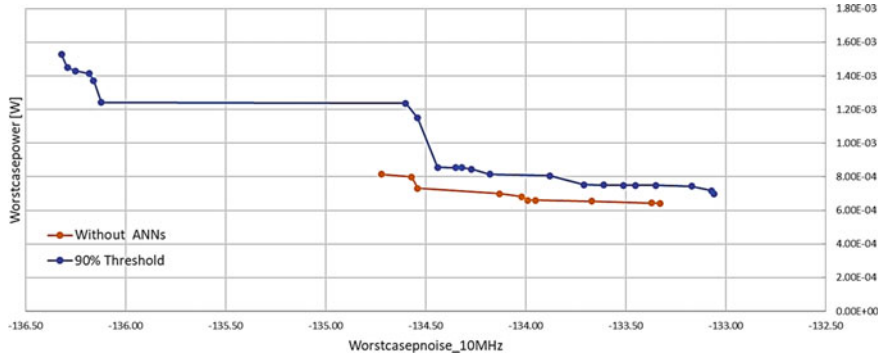


Fig. 3.29 POF obtained with 90% threshold

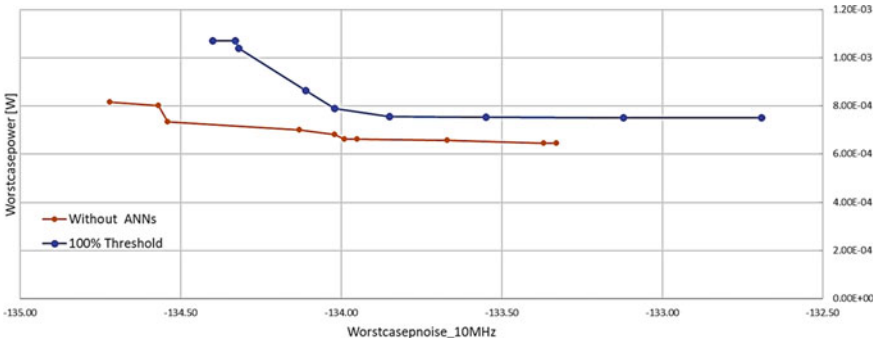


Fig. 3.30 POF obtained with 100% threshold

be evaluated appropriately. Having tested different values for the threshold, the best results were generated with a threshold of 75%, which was the value used for the following tests.

3.4.4 Analysis of the Points Fed to the Simulator

Having tested several values for the threshold, the points that were forwarded to the simulator by the ANNs were studied to have a deeper understanding of the differences between what the ANNs predicted and the outcome of the simulator. This verification also serves to understand if some PVT corner was harder to predict or evaluate. Hence, for an optimization produced by a value of 75% threshold, the points that were sent to the simulator along the optimization cycle were analyzed.

Starting with the points obtained from the first generation, 15 points were fed to the simulator. From those, the ANNs predicted that all those points would converge in all PVT corners, apart from three points where each would fail in a corner. Two of these points were predicted to fail to converge for the ff corner, one in the first tuning mode, one in the second, and the third point was predicted to not converge for the ss corner in the second tuning mode. However, the simulator did not manage to converge for more than only the predicted three sizing-corner pairs. Figure 3.31 shows the distribution of non-converged simulations for each corner-tuning mode combination for the first generation of points.

Advancing to the 75th generation, there were fewer differences in the output of the ANNs and the simulator. Analyzing the 23 points that the ANNs fed to the simulator, only one point was predicted to not converge, in the ss corner for the second tuning mode, whereas the simulator failed to simulate for three different corners, and in the case of the ss corner in the second tuning mode, one additional point than the one predicted failed to converge, as shown in Fig. 3.32.

Finally, in the 150th generation, the ANNs predicted that all the generation's 24 points should be simulated, with zero cases of corners not converging, whereas the

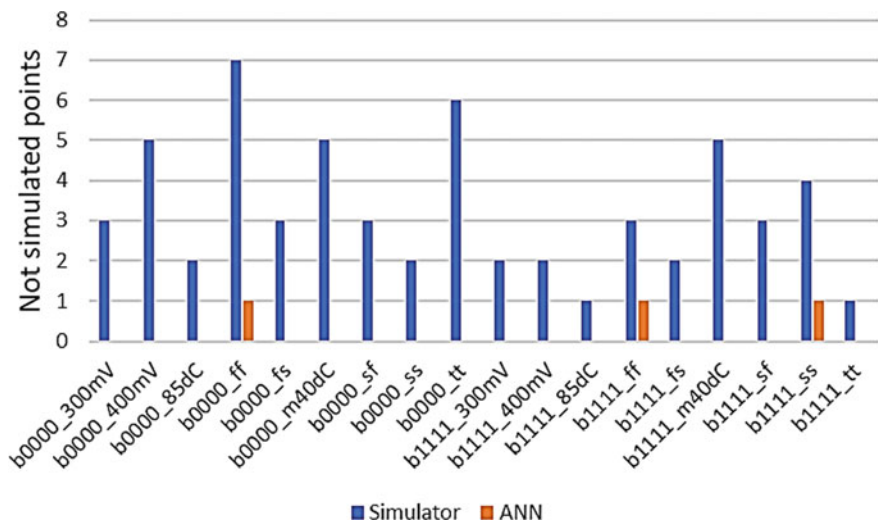


Fig. 3.31 Difference between the prediction of the classification ANN and the output of the simulator for the first generation

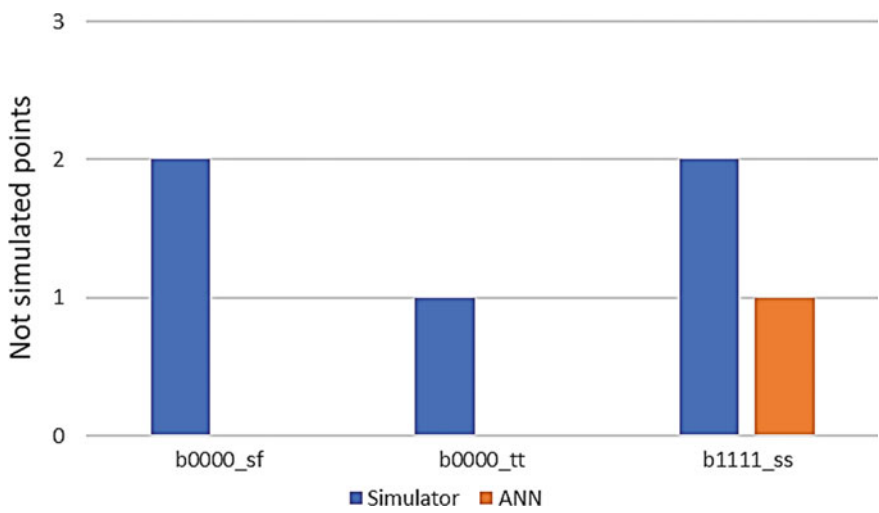


Fig. 3.32 Difference between the prediction of the classification ANN and the output of the simulator for the 75th generation

simulator failed to converge in seven of those points and seven different corners. The ss corner in the second tuning mode was the corner-tuning combination that registered the most non-convergence, as observed in Fig. 3.33.

An earlier analysis of the dataset made it possible to obtain the number of points that converged for each PVT corner. Table 3.14 showcases these results, and it

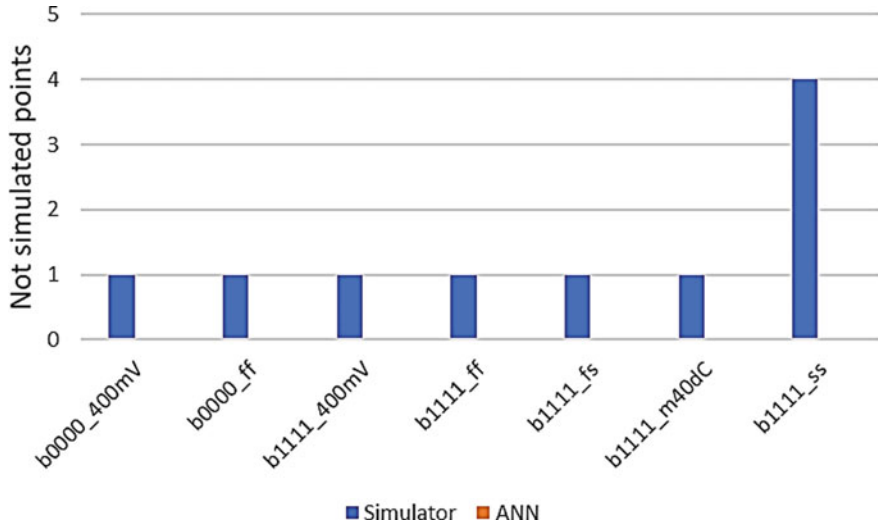


Fig. 3.33 Difference between the prediction of the classification ANN and the output of the simulator for the 150th generation

Table 3.14 Percentage of converged points for each corner

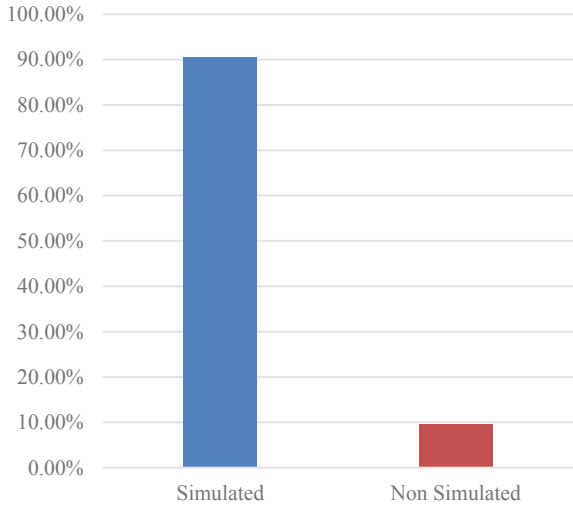
	TT	FF	FS	SF	SS	300mV	400mV	m40dC	85dC
b0000	87,462	87,200	87,881	88,407	79,130	82,206	89,449	87,094	86,678
%Total	94.95	94.66	95.40	95.97	85.90	89.24	97.11	94.55	94.10
b1111	85,143	84,631	86,717	82,049	52,680	76,735	89,021	82,377	87,513
%Total	92.43	91.88	94.14	89.07	57.19	83.30	96.64	89.43	95.00

can be concluded that the corner ss, for both tuning modes, is the one with the highest rate of non-convergence, meaning that this is the most challenging corner to retrieve performance values from. Additionally, this happens more often in the second tuning mode, reaching almost half of the cases. This explains why the ANNs predict that corner fail more often. Furthermore, it also explains why the simulator often fails to converge for that corner, as the data suggests it to be the hardest corner to converge. Moreover, Table 3.14 also explains why the ANNs tend to misclassify points as convergent since the dataset presents more convergence scenarios than non-convergence.

3.4.5 Plug-and-Play Class C/D VCO 2.3 GHz-to-2.5 GHz

A new test was made to prove the efficacy of the ANNs within the same circuit but with slight operational differences. A new optimization was made, and the same

Fig. 3.34 Ratio of points discarded for the new specifications



ANNs trained for the previous specifications were reused in a new setup where the specifications for the circuit were altered. For this setup, the circuit was meant to operate at 2.4 GHz, having a range of frequency from 2.3 to 2.5 GHz as opposed to the previous 3.5–4.8 GHz conditions, and the constraints of the phase noise were changed to a stricter 5 dBc/Hz.

The new optimization was run using a population of 256 elements for 200 generations. The ratio of points simulated and discarded was again recorded and is illustrated in Fig. 3.34. The number of points discarded was low, only 9.51.

Even with a low number of points discarded, the optimal points obtained could present good values, so the POF was retrieved and compared with the one obtained without using the ANNs. Analyzing the results, one can observe that the POF obtained while using the ANNs has better results in terms of phase noise, as the solutions are always lower than -138.50 . However, regarding power consumption, the minimum obtained value was $6.00\text{E}-04$ W, whereas the power consumption values obtained without using ANNs never surpassed the $5.00\text{E}-04$ W mark, as observed in Fig. 3.35.

Ultimately, the results are acceptable, with a reduction of almost 10% of the time. It corresponds to a reduction of almost 3 days since the optimization took about 26 days to complete without using ANNs. In this case, the POFs are distanced and non-dominant, indicating that the optimizations (which are random) converged to different parts of the design space. Still, the filter did work when facing different specifications.

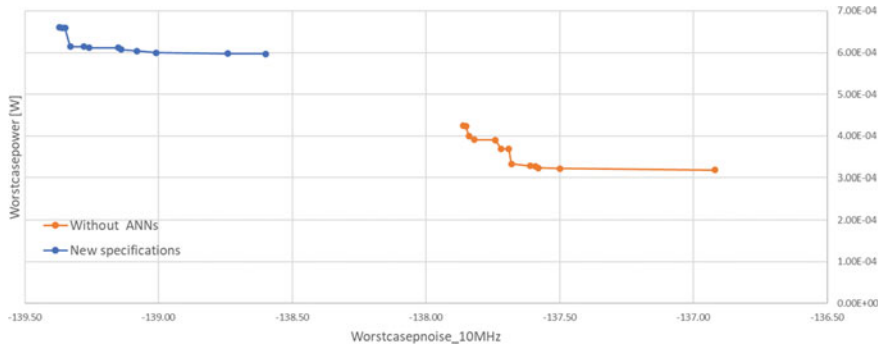


Fig. 3.35 POF obtained with the new specifications

3.4.6 Plug-and-Train Ultralow-Power Class B/C VCO

Finally, the ANNs were trained in a second circuit, the Class-B/C VCO [24]. The same approach was used as the one discussed in Sect. 3.4, so the optimization was performed, and the number of simulated points was registered. For this circuit, the configurations were changed, as the ones used in the reference optimization cycle with no ANNs were also modified. So a population of 256 points was used, and its optimization was performed for 100 generations. As Fig. 3.36 shows, from the total points generated, 17.27% were discarded. This number is close to the 19.65% obtained for the first circuit, the Class-C/D VCO, with a value of also 75% threshold. The resulting POF, shown in Fig. 3.37, was obtained and compared with the reference POF obtained from the optimization without ANNs. The results are promising, as the values of power obtained are lower, reaching a minimum of about $1.50\text{E}-04$ W, whereas the reference values are all greater than around $2.00\text{E}-04$ W. Regarding phase noise, the results are very similar, with three points reaching values below any recorded in the reference optimization.

3.5 Conclusions and Future Research Directions

This work proposed a seamless filter/helper for exhaustive PVT-inclusive RF sizing optimizations. While the savings in terms of computational effort are conservative when compared with methodologies that entirely replace the simulator with a model, here, simulator-grade accuracy is kept throughout the whole process. Not only are irrelevant solutions bypassed, but additionally, by providing an accurate *guess* to the simulator, optimization results are improved.

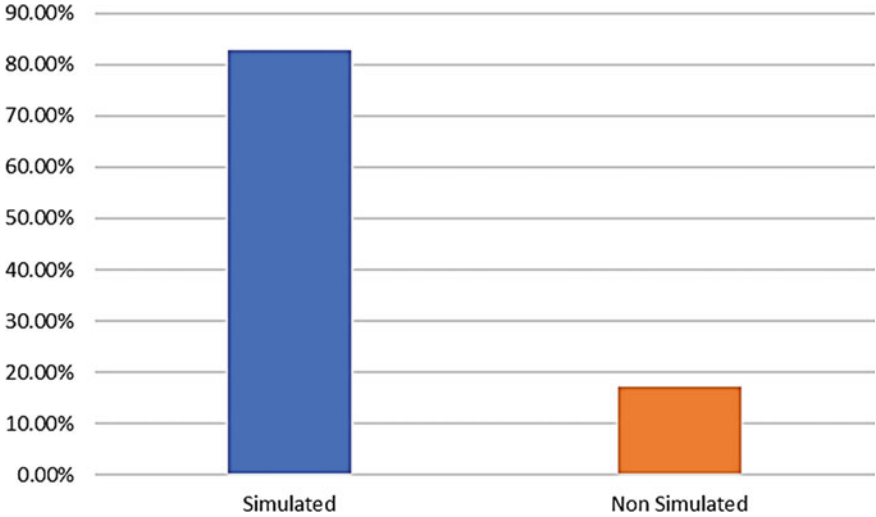


Fig. 3.36 Ratio of points discarded for the new circuit

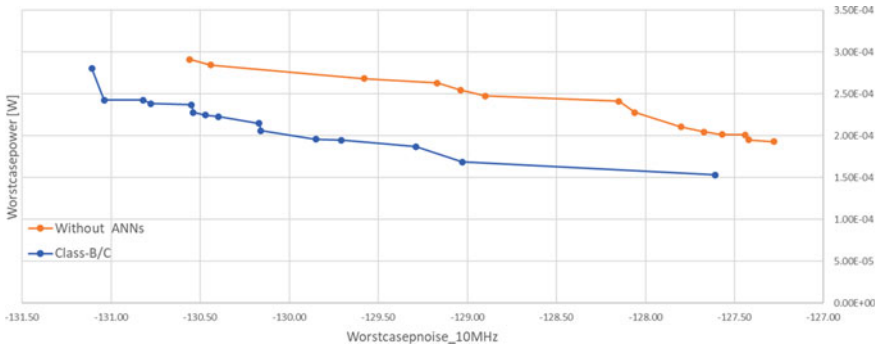


Fig. 3.37 POF obtained for the class-B/C

3.5.1 Conclusions

The work here exhibited showed an approach to optimize the sizing of analog IC circuits with the help of two models, one used for classification and another trained for regression. They both proved valuable, as the classification and regression ANNs discard unwanted solutions, and the use of the regression ANN makes the optimization more effective.

Two circuits were studied, and their design was optimized. The first one was a Class-C/D VCO, where thanks to the convergence filters, it was possible to reduce almost 20% of the optimization time without compromising the results. The obtained POF was even better in some cases than the one obtained via more traditional means.

For the same circuit, the ANNs also proved to work for different specifications, with less efficiency, as the time saved was close to 10%, and the solutions obtained better in one of the optimization metrics but worse in the other.

A second circuit, an Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO, was analyzed and optimized using the same ANN architecture used for the previous circuit, the Class-C/D. A comparison with the unaltered optimization cycle was made, following the same process used for the previous circuit, and the results obtained were discussed and compared with the reference. The results were marginally different, proving there is no need to spend time optimizing a new ANN architecture and emphasizing the optimized architecture's generalization capabilities. The design of this circuit was successful, reducing almost as much time as for the first circuit, and its POF showed very competitive results.

3.5.2 Future Work

This work showed that using ANNs reduces the time required to design an analog IC, even if the time necessary for their training is considered. Having this thought in mind, it is possible to take a step further.

First, even though the ANN architectures have demonstrated a high generalization towards different types of VCOs, the time required to train them must be considered. A future objective would be to reduce this time, which would be possible with additional computational resources. More, the ANNs hyperparameter optimization of the ANNs was performed empirically, requiring human input. This process could be optimized in the future via automatic algorithms such as Bayesian optimization, possibly reducing the time required.

Furthermore, future implementations could consider an online training approach where both ANNs were trained during the first few generations of the optimization cycle, as opposed to the offline approach used in this work, where the model was pre-trained, requiring a previously built dataset. Following this online approach, the first few generations of the optimization cycle would follow the unaltered algorithm as the ANNs were trained in parallel. Once some condition was fulfilled, such as the minimum number of generations or minimum ANN performance, the models would be inserted in the optimization loop, filtering non-converging points and proposing an accurate frequency *guess*.

One challenge with this approach is the relatively low amount of data the ANNs would have to train before deployment such that, overall, their introduction would still be worth it. However, the MAPE evolution throughout the optimization cycle suggests that the population quickly converges to a relatively small region of space. Thus, it is likely that a relatively small of data is enough for the models to reach a significant performance threshold.

References

1. Afacan E, Dündar G (2019) A comprehensive analysis on differential cross-coupled CMOS LC oscillators via multi-objective optimization. *Integr VLSI* 67:162–169
2. Passos F et al (2018) Enhanced systematic design of a voltage controlled oscillator using a two-step optimization methodology. *Integr VLSI* 63:351–361
3. Liao T, Zhang L (2017) Parasitic-aware GP-based many-objective sizing methodology for analog and RF integrated circuits. In: ASPDAC
4. Passos F et al (2020) Ready-to-fabricate RF circuit synthesis using a layout- and variability-aware optimization-based methodology. *IEEE Access* 8:51601–51609
5. Martins R et al (2020) Design of a 4.2-to-5.1 GHz ultralow-power complementary class-B/C hybrid-mode VCO in 65-nm CMOS fully supported by EDA tools. *TCAS-I* 67(11):3965–3977
6. Afacan E, Lourenço N, Martins R, Dündar G (2021) Review: machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test. *Integr VLSI* 77:113–130
7. Suissa A et al (2010) Empirical method based on neural networks for analog power modeling. *IEEE TCAD* 29(5):839–844
8. Wolfe G, Vemuri R (2003) Extraction and use of neural network models in automated synthesis of operational amplifiers. *IEEE TCAD* 22(2):198–212
9. Alpaydin G, Balkir S, Dündar G (2003) An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Trans Evol Comput* 7(3):240–252. <https://doi.org/10.1109/TEVC.2003.808914>
10. Liu H, Singhee A, Rutenbar A, Carley LR (2002) Remembrance of circuits past: macromodeling by data mining in large analog design spaces. In: Proceedings 2002 design automation conference, pp 437–442
11. Lourenço N et al (2019) Using polynomial regression and artificial neural networks for reusable analog IC sizing. In: 16th International conference on synthesis, modeling, analysis and simulation methods and applications to circuit design, pp 13–16, July 2019
12. Zhu K et al (2019) Genius route: a new analog routing paradigm using generative neural network guidance. In: Proceedings of the ICCAD
13. Guerra D, Canelas A, Póvoa R, Horta N, Lourenço N, Martins R (2019) Artificial neural networks as an alternative for automatic analog IC placement. In: International conference on SMACD, Lausanne, Switzerland, July 2019
14. Gusmão A, Passos F, Póvoa R, Horta N, Lourenço N, Martins R (2020) Semi-supervised artificial neural networks towards analog IC placement recommender. In: IEEE International symposium on circuits and systems, Seville, Spain, Oct 2020
15. Gusmão A, Horta N, Lourenço N, Martins R (2022) Scalable and order invariant analog integrated circuit placement with attention-based graph-to-sequence deep models. In: Expert systems with applications. Elsevier, Amsterdam
16. Gusmão A, Póvoa R, Horta N, Lourenço N, Martins R (2022) DeepPlacer: a custom integrated OpAmp placement tool using deep models. In: Applied soft computing, vol 115. Elsevier, Amsterdam, 108188
17. Gusmão A, Horta N, Lourenço N, Martins R (2021) Late breaking results: attention in Graph2Seq neural networks towards push-button analog IC placement. In: ACM/IEEE design automation conference (DAC), San Francisco, USA, Dec 2021
18. Andraud M, Stratigopoulos H, Simeu E (2016) One-shot non-intrusive calibration against process variations for analog/RF circuits. *IEEE TCAS-I Reg Pap* 63(11):2022–2035
19. İslamoğlu G, Çakıcı T, Afacan E, Dündar G (2019) Artificial neural network assisted analog IC sizing tool. In: International conference on SMACD, July 2019
20. Çakıcı T et al (2020) Improving POF quality in multi objective optimization of analog ICs via deep learning. In: ECCTD, pp 1–4, Sept 2020
21. Hakhamaneshi K et al (2019) BagNet: Berkeley analog generator with layout optimizer boosted with DNNs. *IEEE/ACM ICCAD*, Nov 2019

22. Brachtendorf HG, Welsch G, Laur R (1995) Fast simulation of the steady-state of circuits by the harmonic balance technique. In: Proceedings of ISCAS'95—international symposium on circuits and systems, vol 2, pp 1388–1391. <https://doi.org/10.1109/ISCAS.1995.520406>
23. Martins R et al (2019) Many-objective sizing optimization of a class-C/D VCO for ultralow-power IoT and ultralow phase-noise cellular applications. *IEEE TVLSI* 27(1):69–82
24. Martins R, Lourenço N, Horta N, Zhong S, Yin J, Mak P-I, Martins RP (2020) Design of a 4.2-to-5.1 GHz ultralow-power complementary class-B/C hybrid-mode VCO in 65-nm CMOS fully supported by EDA tools. *IEEE Trans Circ Syst I Reg Pap (IEEE TCAS-I)* 67(11):3965–3977
25. Ertam F, Aydın G (2017) Data classification with deep learning using Tensorflow. In: 2017 International conference on computer science and engineering (UBMK), pp 755–758
26. Lee H, Song J (2019) Introduction to convolutional neural network using Keras; An understanding from a statistician. *Commun Stat Appl Methods* 26(6):591–610
27. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15:1929–1958
28. Greff K, Srivastava R, Koutník J, Steunebrink B, Schmidhuber J (2015) LSTM: a search space odyssey. *IEEE Trans Neural Networks Learn Syst* 28. <https://doi.org/10.1109/TNNLS.2016.2582924>
29. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. *J Mach Learn Res Proc Track* 9:249–256

Chapter 4

Process, Voltage and Temperature Corner Performance Estimator Using ANNs



4.1 Contributions

Real-world applications demand RF IC blocks' robustness, and to ensure it, automatic sizing has moved towards exhaustive PVT-inclusive optimizations. In such scenarios, the relevant performances of each candidate sizing solution are found by simulating the circuit under different operating modes as well as fabrication dispersions and voltage/temperature variations, escalating the time required for optimization. In this chapter, DL is used to improve RF IC sizing automation, and the major contributions are listed as follows:

- Previous works in this research field [3–5, 14, 15] focus on sizing on nominal conditions, and still, significant errors are observed in the estimated circuit performances, hindering its use for complex PVT-inclusive optimizations. Here, ANNs tailored for exhaustive PVT-inclusive RF IC sizing optimization problems are proposed;
- The proposed controlled PVT regressor incorporates the nominal performance figures obtained via circuit simulation in the input layer of its ANNs. Therefore, the model becomes flexible to operate in different regions of the performance space. Ultimately the same model can be used for optimizations with a completely different set of targets, i.e., generalizing beyond training data;
- Instead of entirely replacing the simulator [3–5, 14, 15], two control phases used in every generation prevent the sizing loop from being misled by inaccurate performance estimates and consequently guided to unrealistic design space regions;
- The proposed controlled PVT regressor is tested on different optimizations of two state-of-the-art voltage-controlled oscillators (VCOs), reducing the workload of the simulator up to 79%, i.e., saving more than 16 days of computational effort while achieving competitive sizing solutions (Fig. 4.1).

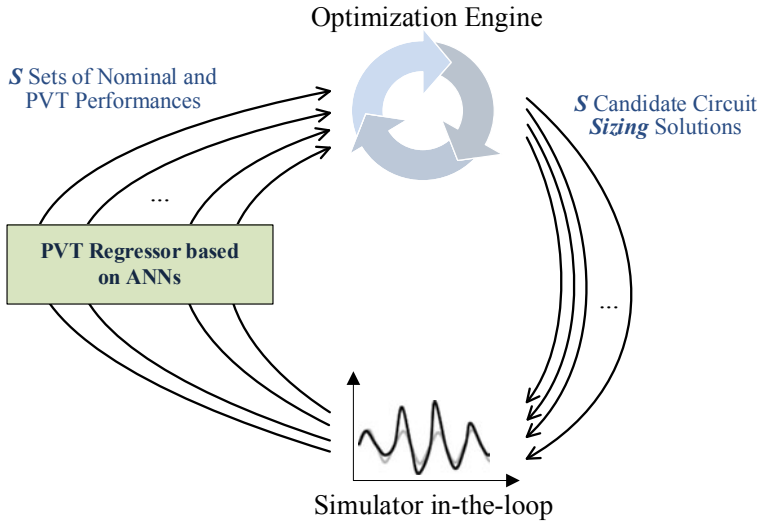


Fig. 4.1 Location of the PVT estimator in a simulation-based optimization loop

4.2 Controlled PVT Regressor Based on Deep ANNs

Recovering the dual-mode class C/D VCO case study [16] introduced in Chap. 2 of this book (Sect. 2.5), a dataset generated using the current problem definition has a total number of 48 features, where 28 are the optimization variables plus 20 performance figures of the simulation in typical (TT) conditions in two different modes, i.e., b_{0000} and b_{1111} , and a total of 160 labels, i.e., the performance figures of the remaining corner variations in the two different tuning modes.

Each ANN of the controlled PVT regressor will estimate the performance figures of a specific corner for a specific tuning mode so that the output layer will have ten neurons. Each ANN will receive, as inputs, the sizing of the circuit, which consists of the same 28 optimization variables and ten performance figures of the simulation in TT conditions for its corresponding tuning mode, which means that the input layer will have a total of 38 neurons. The number of hidden layers and number of nodes per hidden layer will be determined in the tuning phase, and a study will be conducted to find the best possible solution to these two hyperparameters. The structure of the ANN implemented for each corner and tuning mode is shown in Fig. 4.2, where a chain of fully connected layers is used. The output of the regression model is a real value, positive or negative, so the output nodes have no activation function. Overall, the structure of the PVT estimator, containing 16 different ANNs, is further detailed in Fig. 4.3.

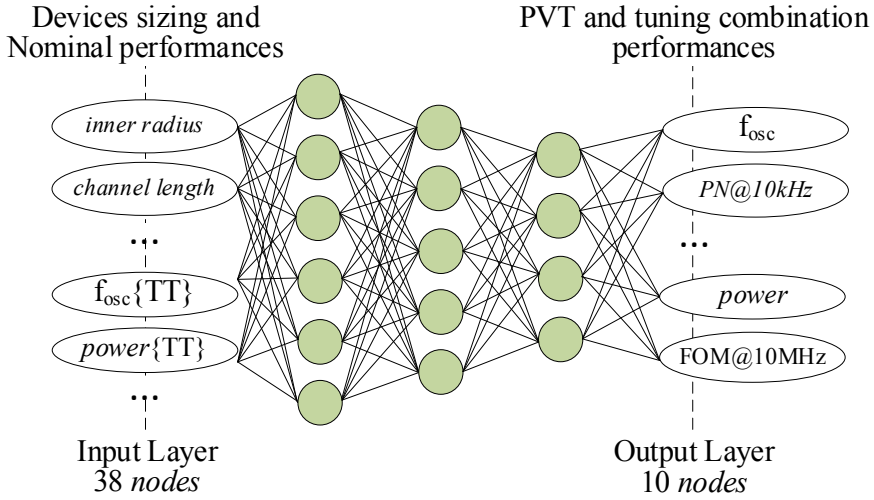


Fig. 4.2 ANN structure for corner FF and tuning mode b_{0000}

4.3 Training the Model in Isolation (Results Pre-integration)

For training the PVT estimator's ANNs, first, the datasets for each ANN were defined along with the necessary pre-processing to obtain the best data for their training phase. Furthermore, the tuning phase of the hyperparameters of the ANNs is described, followed by the showcase of the final model structure. Finally, the test results of one of the hyperparameters of the ANNs are described. These results show that the ANN can achieve highly accurate results on unseen data and are considered adequate for the remaining experiments. In order to do this, the dataset of the complex dual-mode class C/D voltage-controlled oscillator (VCO) circuit, previously defined in Chap. 2, will be used.

Again the models were implemented in Python, using both Tensorflow [17] and Keras [18] as ML libraries. The starting point of the ANN architecture is the one described in Fig. 4.2, where the output layer contains 10 nodes, one for each performance parameter of a certain combination of tuning mode and PVT corner variation. The optimization was performed for 9 testbench variations (TT and 8 PVT corners) and 2 tuning modes; thus, 16 different ANNs will be required.

4.3.1 Dataset Processing

The dataset contains 92,115 data entries composed of, as previously described, 48 features, where 28 represent the optimization variables and the other 20 represent the

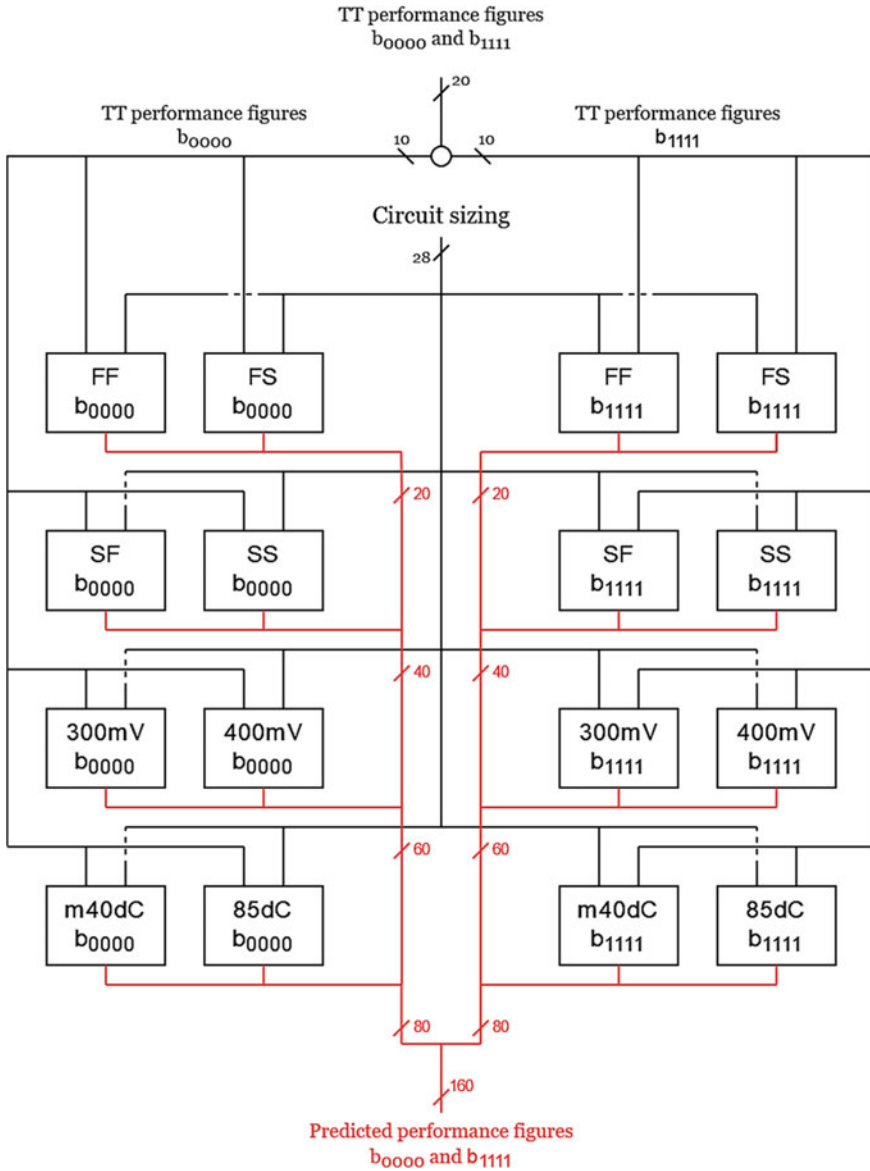


Fig. 4.3 Detailed diagram of the PVT estimator

TT performance figures, 10 for each tuning mode. As for labels, the dataset contains 160 performance figures of the remaining PVT corner variations in two different tuning modes. For each different ANN, it is only needed the performance figures of one combination of corner variations and tuning mode, so firstly, the dataset had to be divided into 16 different datasets where each dataset represents a different corner

combination. Only the TT performance figures representing the same tuning mode as the labels are kept to increase model accuracy, so the final dataset structure only contains 38 features and 10 labels.

Some data entries have *null* values on the features and/or labels, representing sizing solutions that the simulator could not produce a meaningful performance figure. These entries had to be removed, along with duplicated rows, from each dataset to provide the best possible data to the ANNs.

The division of the original dataset into smaller datasets and the removal process explained before are depicted in Tables 4.1 and 4.2, where some entries of the original dataset and dataset for FF b_{0000} are shown, respectively.

Finally, the outliers present in each dataset must be removed. For each performance figure, the 1% lowest and highest values were cut from the dataset alongside their entire row of data. The final sizes of the dataset for each PTV corner ANN can be found in Table 4.3. Finally, all 16 datasets were randomized and split into two datasets: the training dataset, which consists of 90% of the original dataset, and the test dataset, which is the remaining 10% of the original dataset. The training dataset will be used to train the ANNs, while the test dataset will be used to test the models. As for the tuning of the ANN architecture, it was only performed in the ANN regarding the corner FF with tuning mode b_{0000} .

Table 4.1 Structure of the original dataset

Entry #	Features (48)			Labels (160)			
	Design variables	Perform. TT b_{0000}	Perform. TT b_{1111}	Perform. FF b_{0000}	...	Perform. m40dC b_{1111}	Perform. 85dC b_{1111}
0	Values ✓	Values ✓	Values ✓	Null	...	Values ✓	Null
1	Values ✓	Values ✓	Values ✓	Values ✓	...	Null	Null
2	Values ✓	Null	Null	Values ✓	...	Null	Null
3	Values ✓	Values ✓	Values ✓	Null	...	Null	Null
...	...	...	...	...	...	...	...
92,114	Values ✓	Values ✓	Values ✓	Values ✓	...	Values ✓	Values ✓

Table 4.2 Structure of the dataset for FF b_{0000}

Entry #	Original entry #	Features (38)		Labels (10)
		Design variables	Performances TT b_{0000}	Performances FF b_{0000}
0	1	Values ✓	Values ✓	Values ✓
1	3	Values ✓	Values ✓	Values ✓
...	...	...	...	...
83,037	92,114	Values ✓	Values ✓	Values ✓

Table 4.3 Dimensions of datasets for the training of the different ANNs

Corner	Tuning mode			
	b_{0000}	% total	b_{1111}	% total
FF	81,377	88.34	79,300	86.09
FS	81,842	88.85	81,200	88.15
SF	82,247	89.29	76,794	83.37
SS	73,456	79.74	48,742	52.91
300mV	77,608	84.25	69,017	74.92
400mV	81,865	88.87	81,342	88.30
m40dC	81,182	88.13	77,103	83.70
85dC	80,707	87.62	82,028	89.05

4.3.2 Feature Engineering

All this raw data has to be pre-processed before using it in the training phase of the model so some feature engineering will be needed. The dataset contains small input device sizes (in the order of nanometers) combined with other optimization variables, which can be simple integers (for example, *ind_nturns*), and with performance figures, which can have large values in the order of gigahertz (for example, *oscillationfrequency*). This combination causes the learning algorithm of the ANN to wrongly compute the weights associated with these small values, almost completely negating their influence on the output. To solve this problem, data normalization will be performed on the entirety of the dataset. Two methods can be used to achieve this: standardization and normalization. Standardization scales the values while considering standard deviation, which is beneficial to reduce the effect of outliers in the data. Normalization scales all values to a fixed range. This scaling does not alter the feature distributions, and because of the decreased standard deviations, the effect of the outliers increases. The expressions for standardization and normalization are shown in (4.1) and (4.2), respectively, where μ represents the mean value and σ the standard deviation.

$$z = \frac{x - \mu}{\sigma} \quad (4.1)$$

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4.2)$$

Although many other feature engineering techniques can be performed on the dataset, the previously mentioned ones will be the most important. After this, the dataset is ready to be fed to the network.

4.3.3 Tuning Hyper-Parameters

With the dataset ready, the ANN's hyper-parameter tuning phase starts. Three metrics were used as evaluation methods for the ANN's training: MSE, defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (4.3)$$

Mean absolute error (MAE), defined by:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - x_i| \quad (4.4)$$

And finally, Mean absolute percentage error (MAPE), defined by:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (4.5)$$

Due to the nature of the values that are trying to be predicted with this ANN, the error between the predicted value and the actual value must be small. Because of this, an error value lower than 1% (for the case of the MAPE) was chosen as a reasonable target to achieve the accuracy of the ANN.

Despite having three different metrics as evaluation methods, the loss function of the ANN throughout the tuning phase was the MSE. It is one of the most popular loss functions in regression problems like the one addressed in this book. A similar approach was used to choose the optimizer, with Adam being the most popular method. Regarding the batch size of the training phase, 128 was chosen, resulting in a mini-batch approach where the batch is small enough to introduce some helpful noise in the training process while diminishing the time increase associated with stochastic gradient descent (i.e., batch size of 1). The normalization range of 1–2 was chosen, over the typical 0–1, due to its influence on MAPE values. If a range including zero was chosen, MAPE values would “*explode*” due to target values close to 0 appearing in the denominator of (4.5).

The first parameters to be determined were: the number of layers, the number of neurons per layer, and the learning rate. Two studies were made to determine the adequate number of layers for the model, the first using 2 hidden layers and the second 3 hidden layers. Table 4.4 presents the parameters and values considered in the 2 hidden layers study.

It is common practice in ANN development that the size of a hidden layer is always equal to or larger than the following hidden layer. Considering this, the different combinations of these parameters were studied, and their results are shown in Table 4.5. The lowest values of each metric (MSE, MAE, and MAPE) at both training and

Table 4.4 Different parameters and corresponding values to study

Size of hidden layer 1	Size of hidden layer 2	Learning rate
200, 320, 440, 560	200, 300, 400, 500	0.00005, 0.0001, 0.0005, 0.001

test phases were highlighted, and achieving 5 of the 6 total lowest metrics, the best combination of these parameters is:

- Hidden layer size 1: 440 neurons
- Hidden layer size 2: 400 neurons
- Learning rate: 0.0001

Given these results, a comparison with 3-hidden layer ANN architecture is carried. In this next study, the learning rate was set to 0.0001, considering it was the best

Table 4.5 Results of hidden layers size/learning rate study. Using activation function of hidden layers: ReLU, dropout rate: 20%

Hidden layers		Learning rate	Training loss			Test loss		
#1	#2		MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE
200	200	1×10^{-3}	2.0271	9.0719	0.5865	2.9237	9.6342	0.6262
		5×10^{-4}	1.6593	8.4255	0.5503	2.6371	8.9969	0.5903
		1×10^{-4}	0.9633	4.9812	0.3263	2.0497	5.6614	0.3736
		5×10^{-5}	1.5778	6.3185	0.4126	2.4236	6.9236	0.4543
320	200	1×10^{-3}	1.6222	8.4173	0.5433	2.5826	9.0382	0.5867
		5×10^{-4}	1.2829	7.7522	0.5101	2.1944	8.4498	0.5579
		1×10^{-4}	0.7502	4.2473	0.2801	1.7580	4.9959	0.3316
		5×10^{-5}	1.1752	5.3409	0.3510	1.9515	5.9199	0.3911
320	300	1×10^{-3}	1.5518	7.4064	0.4838	2.4660	7.9911	0.5243
		5×10^{-4}	0.9664	6.0647	0.3905	1.8117	6.7411	0.4368
		1×10^{-4}	0.6854	4.0345	0.2649	1.6603	4.8014	0.3176
		5×10^{-5}	1.5619	6.6477	0.4345	2.5805	7.3492	0.4827
440	200	1×10^{-3}	0.9685	5.9367	0.3893	1.7024	6.5298	0.4304
		5×10^{-4}	0.9217	5.7347	0.3669	1.7494	6.4228	0.4139
		1×10^{-4}	0.6052	4.0061	0.2625	1.4736	4.7756	0.3149
		5×10^{-5}	1.2531	6.3213	0.4136	2.2113	7.0737	0.4648
440	300	1×10^{-3}	2.0921	8.0931	0.5290	3.4045	8.7485	0.5760
		5×10^{-4}	0.9115	6.3530	0.4114	1.7763	7.0825	0.4612
		1×10^{-4}	0.6586	4.0705	0.2677	1.4629	4.7683	0.3155
		5×10^{-5}	1.2429	6.0380	0.3935	2.1168	6.7248	0.4407

(continued)

Table 4.5 (continued)

Hidden layers		Learning rate	Training loss			Test loss		
#1	#2		MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE
440	400	1×10^{-3}	1.1995	6.0374	0.4000	2.1117	6.6899	0.4457
		5×10^{-4}	0.6916	4.7748	0.3087	1.5629	5.4907	0.3578
		1×10^{-4}	0.5372	3.5950	0.2344	1.5345	4.4429	0.2919
		5×10^{-5}	1.1603	5.4496	0.3574	2.3146	6.2074	0.4093
560	200	1×10^{-3}	1.0060	6.2592	0.4213	1.9085	6.9229	0.4667
		5×10^{-4}	0.7292	5.0753	0.3370	1.5336	5.8173	0.3874
		1×10^{-4}	0.5895	3.8783	0.2558	1.5572	4.6842	0.3110
		5×10^{-5}	0.9646	5.0136	0.3267	1.9388	5.7902	0.3798
560	300	1×10^{-3}	0.9744	5.2125	0.3425	2.1092	5.9488	0.3937
		5×10^{-4}	0.7596	5.2312	0.3366	1.7456	5.9887	0.3887
		1×10^{-4}	0.5920	3.9961	0.2601	1.4898	4.8121	0.3157
		5×10^{-5}	0.8541	4.6235	0.3033	1.6815	5.3325	0.3516
560	400	1×10^{-3}	5.8195	14.5366	0.9489	6.4470	14.9806	0.9800
		5×10^{-4}	0.8302	5.4530	0.3566	1.6963	6.1971	0.4072
		1×10^{-4}	0.5769	3.8253	0.2504	1.6146	4.6759	0.3085
		5×10^{-5}	0.8927	4.9385	0.3201	1.7218	5.6835	0.3707
560	500	1×10^{-3}	1.5522	7.6797	0.5064	2.4262	8.3103	0.5505
		5×10^{-4}	0.5916	4.2945	0.2834	1.4161	5.0691	0.3365
		1×10^{-4}	0.6000	4.5236	0.3017	1.5613	5.3601	0.3586
		5×10^{-5}	0.8585	5.3343	0.3411	1.7989	6.1322	0.3955

value of the previous study. Table 4.6 shows the different parameters and the values that were considered. This study follows the same rule as the previous one regarding the sizes of the hidden layers, and its results are shown in Table 4.7.

When comparing the results of the two studies, it is clear that there is no improvement when increasing the number of hidden layers. The best error results with 3 hidden layers are 18–30% higher than the best results with 2 hidden layers. Considering this fact, there is no need to increase the number of hidden layers of the ANN, so no further study was required. These values (2 hidden layers with 440 and 400 neurons each, respectively, and a learning rate of 0.0001) will be used for the remainder of the tuning phase.

Table 4.6 Different parameters and corresponding values to study

Size of hidden layer 1	Size of hidden layer 2	Size of hidden layer 3
200, 320, 440	200, 300, 400	200, 300, 400

Table 4.7 Results of hidden layers size study using activation function of hidden layers: ReLU, learning rate: 0.0001 and dropout rate: 20%

Hidden layers			Training loss			Test loss		
#1	#2	#3	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE
200	200	200	1.9722	8.7683	0.5709	3.1234	9.4508	0.6186
320	200	200	0.9191	5.8708	0.3796	2.0354	6.7268	0.4385
320	300	200	1.5984	7.7507	0.5039	2.8096	8.5406	0.5586
320	300	300	0.9735	6.1391	0.3957	2.0810	7.0175	0.4560
440	200	200	0.7186	4.4182	0.2897	1.7230	5.2482	0.3465
440	300	200	0.9799	5.9074	0.3852	1.9675	6.6465	0.4361
440	300	300	0.8321	5.6166	0.3613	1.8117	6.4913	0.4210
440	400	200	1.0870	6.4679	0.4147	2.1437	7.2971	0.4718
440	400	300	1.1155	6.7236	0.4337	2.4101	7.6854	0.4997
440	400	400	0.8439	5.5525	0.3570	1.8376	6.4363	0.4172

The following parameter to be tuned is the activation function of the hidden layers. For the hidden layers of the ANN, one should use non-linear activation functions to increase its capability to model highly complex relationships between its input and output. The following functions will be experimented with in the model, and their results will be compared to choose the most adequate for this methodology.

The sigmoid function, shown in Fig. 4.4 (top), can map the input values of a neuron to an output range of]0, 1[. It prevents cases where the neuron’s output reaches very high values, which can happen when taking the unbounded linear output of the neurons. Another quality of this function is its sensibility to input changes in the region near $z = 0$, which results in good separation of data. However, the responsiveness of the function starts to decrease when dealing with bigger and smaller input values, reaching almost a gradient value of 0. Therefore, the model can no longer update properly. One of the most popular activation functions is the rectified linear unit (ReLU) function [19], shown in Fig. 4.4 (bottom-left), computed through $f(z) = \max(0, z)$. The output value of the ReLU function is equal to the input value for inputs greater than 0 and 0 for all the other values. Due to its simpler formula, this non-linear function requires less computational power than the sigmoid function. However, the function is not bounded for positive input values making it susceptible to exploding output values. Despite this, it solves the low gradient value encountered in the sigmoid function for large positive values. The main problem appears in the negative input value region, where the gradient equals 0, which will stop any network update from happening (regarding the corresponding neuron). This is called the dying ReLU problem. To solve it, some variations to the ReLU were developed, such as the leaky ReLU [20] or the exponential linear unit (ELU) [21]. The ELU introduces a smooth derivate for the negative input values and is determined by (4.6), where α is a parameter to be tuned. This function is depicted in Fig. 4.4 (bottom-right).

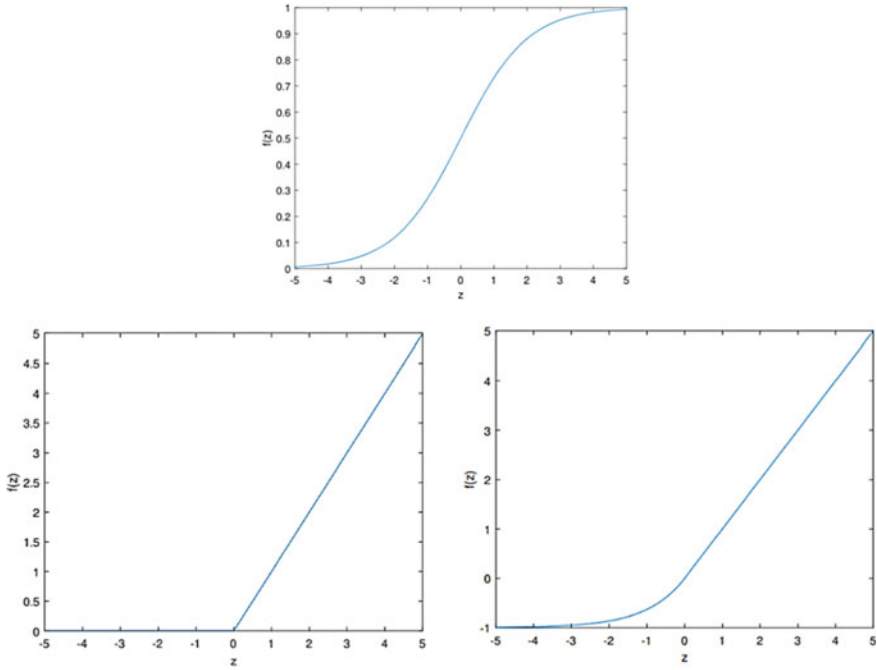


Fig. 4.4 Sigmoid (top), ReLU function (bottom-left) and ELU function (bottom-right)

$$f(z) = \begin{cases} \alpha(e^z - 1) & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases} \quad (4.6)$$

The results of using sigmoid, ReLU, leaky ReLU, and ELU as activation functions are shown in Table 4.8. As can be seen, by the highlighted values, the activation function that generates the lowest error in almost all cases is the ReLU function, proving why it is one of the most popular activation functions when working with ANNs nowadays [19]. Leaky ReLU has the best MSE test loss of all 4 activation functions, but the remaining values fall short of the ReLU. Regarding the sigmoid and ELU functions, these demonstrated weak results when compared with the other 2 functions, getting values 2 times worse in all metrics. This study suggests that the best activation function for this work is the ReLU function, which will be the function used in the final model.

One of the main problems when training an ML model is overfitting. It refers to a model that fits the training data too well, i.e., the model learns the detail and noise in the training data to the point that it negatively impacts the model's performance on new unseen data. Random fluctuations and noise present in the training data do not apply to new data, so it negatively affects the ability of the model to generalize. A regularization technique can be used when training the ANN to solve this problem [22, 23]. Dropout is one popular regularization technique because it solves two

Table 4.8 Results of activation function study. Using 2 hidden layers of sizes 440 and 400, learning rate: 0.0001 and dropout rate: 20%

Activation function	Training loss			Test loss		
	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE
Sigmoid	3.8048	10.4705	0.6847	4.2671	10.8543	0.7120
ReLU	0.5372	3.5950	0.2344	1.5345	4.4429	0.2919
Leaky ReLU	0.8000	4.2949	0.2804	1.4946	4.8755	0.3204
ELU	2.8294	8.4947	0.5537	3.4148	8.9600	0.5855

Table 4.9 Results of dropout rate study using 2 hidden layers of sizes 440 and 400, learning rate: 0.0001 and activation function of hidden layers: ReLU

Dropout rate (%)	Training loss			Test loss		
	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE
0	1.6501	8.4134	0.5456	2.3367	8.8968	0.5786
5	0.5724	3.6952	0.2417	1.6456	4.6661	0.3074
10	0.5346	3.8489	0.2511	1.6371	4.7203	0.3105
20	0.5372	3.5950	0.2344	1.5345	4.4429	0.2919
30	0.7887	4.8243	0.3167	1.7034	5.5186	0.3647

crucial problems in an ANN: it prevents overfitting of the model and provides a method of approximately combining different ANN architectures. Dropout consists in removing some neurons temporarily from the ANN along with its incoming and outgoing connections during training.

The dropout rate, which until now was set to 20%, is the last parameter to be tuned. The different values and corresponding results are shown in Table 4.9. Analyzing the results, dropout rates of 5, 10, and 20% show the best and relatively similar values between them, while the values for a dropout rate of 0 and 30% show a decrease in the accuracy of the ANN. The worst results come from no dropout, which reveals the necessity of this regularization technique. A dropout rate of 20% was chosen for the final model, given that the ANN presents the best results with this value.

4.3.4 Final Model Details

All the parameters of the ANN are now tuned to achieve a good performance. A model summary is shown in Table 4.10, along with the metrics of the training phase at each epoch. The evolution of the loss and error functions in the training and validation set is shown in Figs. 4.5 and 4.6.

Table 4.10 Summary of ANN

Parameter	Value
Input layer size	38
Hidden layer 1 size	440
Hidden layer 2 size	400
Output layer size	10
Loss function	Mean squared error
Optimizer	Adam
Batch size	128
Hidden layers activation function	ReLU
Learning rate	0.0001
Dropout rate	20%
Training epochs	300
Validation split	20%

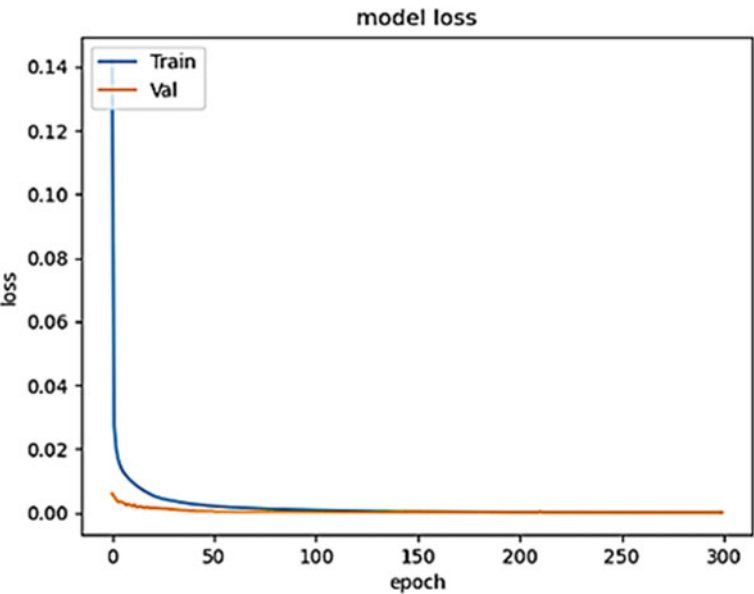


Fig. 4.5 MSE (loss) function

The final values of the 3 metrics for all 16 ANNs are shown in Table 4.11. As can be seen, by the MAPE values of both training and test loss, the final model in all ANNs presents better performance than the initial goal of 1% error.

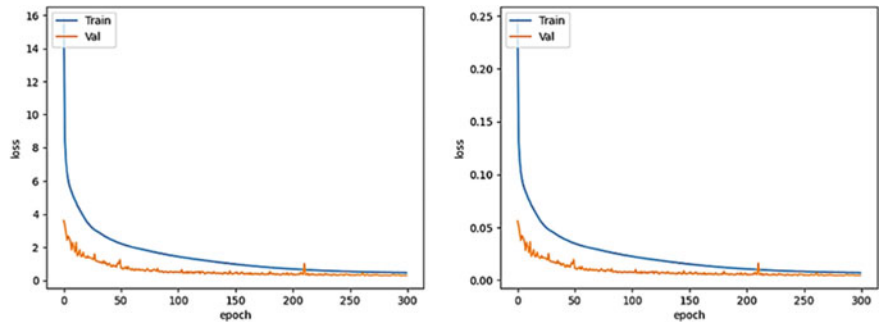


Fig. 4.6 MAPE (left) and MAE (right)

Table 4.11 Final model metric values

Tuning mode	Corner	Training loss			Test loss		
		MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE	MSE ($\times 10^{-4}$)	MAE ($\times 10^{-3}$)	MAPE
b_{0000}	FF	0.5346	3.5950	0.2344	1.5345	4.4429	0.2919
	FS	0.5046	3.7093	0.2438	1.5623	4.3444	0.2873
	SF	0.7096	3.7298	0.2441	1.2777	4.2840	0.2810
	SS	0.9830	4.3566	0.2833	1.6931	4.9340	0.3236
	300mV	0.7763	5.3615	0.3524	1.5838	5.9087	0.3919
	400mV	0.6531	4.5662	0.3061	1.0625	4.9738	0.3339
	m40dC	0.6195	4.0734	0.2665	1.1569	4.3965	0.2890
	85dC	0.6881	4.4254	0.2965	1.1435	4.8061	0.3237
b_{1111}	FF	1.2872	5.4180	0.3638	2.3003	5.9296	0.3987
	FS	0.7244	4.0943	0.2727	1.7098	4.7493	0.3163
	SF	0.8058	4.3929	0.2858	1.7478	5.0881	0.3324
	SS	1.1053	7.8992	0.5777	2.8484	11.5884	0.8384
	300mV	1.8979	7.0264	0.4585	3.4236	8.1042	0.5304
	400mV	1.7348	7.4566	0.4828	3.1930	8.4154	0.5486
	m40dC	5.9536	9.3559	0.6163	8.4010	10.8045	0.7134
	85dC	1.0614	4.5612	0.3037	1.7748	5.1120	0.3418

4.3.5 Test Results

In order to check the performance of the ANN, the prediction errors of 4 different performances were obtained for the points that belong to the test. These performances are: Oscillation frequency; Power; Phase noise in 10 MHz; Figure of Merit in 10 MHz. All 4 of these performances were obtained for corner FF in mode b_{0000} . In Tables 4.12,

4.13, 4.14, and 4.15 the results of the predictions are presented, where 5 different sets of results are shown:

- Best: 5 lowest MAPE values;
- 25th quantile value and the next 4 points (in order of MAPE);
- Median value and the next 4 points (in order of MAPE);
- 75th quantile value and the next 4 points (in order of MAPE);
- Worst: 5 highest MAPE values.

As observable in the results, the ANN can achieve highly accurate results. In all four performances, the best results are almost perfect predictions of the actual value with errors of 0.0005%. Another great indicator of the performance of the ANN is the quartile values. The worst 25th quantile value of the 4 performances is an error of 0.15%, which indicates that 25% of all the predictions have a MAPE lower or equal for that performance (phase noise for this case). The medians of all 4 performances also show evidence of an accurate model ANN. The worst median value, also belonging to the phase noise performance, shows a slight error of 0.28% between prediction and real value. The same can be said for the 75th quantile values, where the worst one is at 0.42% for the phase noise. Once again, these results show the powerful prediction capabilities of the model as it states that 75% of the points predicted to have 0.42% or lower error values. Finally, the worst MAPE values are the extreme situations that the model encountered, precisely, points that belong to a range of values very different than the most representative ones on the dataset. This can be seen, for example, in the worst MAPE values of the oscillation frequency where real values correspond to 1.6, 7.3, 7.2, and 9.9 GHz, which correspond to values very different than the normal band of frequency of the dataset (3.5-to-4.8 GHz).

4.4 In-the-Loop Integration

This section provides a clear explanation of the integration in the AIDA [24] loop of the models developed is presented, and the results of 3 different optimizations using the modified AIDA loop will be shown and discussed. With the ANNs for each corner and tuning mode tuned and ready to be used, the next step of this work is to integrate the PVT estimator into the AIDA loop. The location of the PVT estimator is presented in Fig. 4.7.

In the first step of the optimization loop, several circuit sizing solutions are proposed by the optimization engine (the number of solutions depends on the population defined). In the original AIDA loop, the simulator evaluates each of these solutions for all TT conditions and PVT corners and outputs all the evaluated performances. The optimization engine receives these evaluations and ranks the solutions (population) according to their compliance with the objectives and constraints set for the current optimization problem. A brief flowchart of one generation of the original AIDA loop is shown in Fig. 4.8.

Table 4.12 Error results of oscillation frequency for corner FF in mode b_0000

Best			25th			Median			75th			Worst		
MAPE ($\times 10^{-4}$)	Real (GHz)	Pred. (GHz)	MAPE ($\times 10^{-2}$)	Real (GHz)	Pred. (GHz)	MAPE ($\times 10^{-1}$)	Real (GHz)	Pred. (GHz)	MAPE ($\times 10^{-1}$)	Real (GHz)	Pred. (GHz)	MAPE	Real (GHz)	Pred. (GHz)
0.01	4.863	4.863	7.52	6.078	5.940	1.63	4.792	4.765	3.06	5.282	5.211	4.94	4.155	4.891
0.68	4.363	4.363	7.53	4.508	4.522	1.63	4.346	4.325	3.06	4.923	4.898	5.26	7.322	6.371
0.82	4.814	4.814	7.53	5.175	5.147	1.63	3.134	3.091	3.06	4.751	4.754	6.01	7.184	6.107
0.98	4.632	4.632	7.53	5.353	5.278	1.63	6.054	5.957	3.06	5.047	5.028	6.16	1.619	2.380
1.04	2.535	2.535	7.54	5.554	5.788	1.63	5.356	5.342	3.06	4.987	4.979	6.51	9.852	8.512

Table 4.13 Error results of power for corner FF in mode b_0000

Best			25th			Median			75th			Worst		
MAPE ($\times 10^{-3}$)	Real (mW)	Pred. (mW)	MAPE ($\times 10^{-2}$)	Real (mW)	Pred. (mW)	MAPE ($\times 10^{-1}$)	Real (mW)	Pred. (mW)	MAPE ($\times 10^{-1}$)	Real (mW)	Pred. (mW)	MAPE	Real (mW)	Pred. (mW)
0.02	4.863	4.863	7.46	1.103	1.088	1.63	1.886	1.835	3.13	1.264	1.262	9.83	1.242	4.891
0.07	4.363	4.363	7.46	1.419	1.435	1.63	2.059	2.107	3.13	2.112	2.096	9.92	1.165	6.371
0.12	4.814	4.814	7.47	1.034	1.041	1.63	3.124	2.868	3.13	1.747	1.722	11.16	1.119	6.107
0.15	4.632	4.632	7.47	1.041	1.033	1.63	1.199	1.187	3.13	1.561	1.562	12.08	0.982	2.380
0.17	2.535	2.535	7.47	1.321	1.381	1.63	1.703	1.661	3.13	1.547	1.533	13.02	2.126	8.512

Table 4.14 Error results of phase noise in 10 MHz for corner FF in mode b_0000

Best				25th				Median				75th				Worst			
MAPE ($\times 10^{-3}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE	Pred. (dBc/Hz)
0.12	133.73	133.73	1.54	132.45	132.48	2.80	134.01	133.90	4.21	131.99	131.88	11.66	133.41	128.99					
0.20	134.14	134.14	1.54	133.30	133.08	2.80	134.42	134.31	4.21	131.29	131.18	11.96	132.68	128.06					
0.25	132.22	132.22	1.54	132.37	132.17	2.80	134.16	133.25	4.21	134.43	134.25	13.84	135.65	130.71					
0.25	132.14	132.14	1.54	132.02	132.01	2.80	129.20	130.29	4.21	134.41	134.31	14.68	124.36	131.26					
0.48	132.69	132.69	1.54	129.12	129.05	2.80	133.54	133.33	4.21	133.24	133.11	16.45	130.22	136.98					

Table 4.15 Error results of FOM in 10 MHz for corner FF in mode b_0000

Best				25th				Median				75th				Worst			
MAPE ($\times 10^{-4}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-2}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE ($\times 10^{-1}$)	Real (dBc/Hz)	Pred. (dBc/Hz)	MAPE	Real (dBc/Hz)	Pred. (dBc/Hz)
0.51	183.41	183.41	7.10	187.70	187.62	1.36	184.86	184.83	185.43	185.22	2.17	185.43	185.22	9.02	162.52	167.61			
0.78	183.44	183.44	7.10	184.86	184.64	1.37	184.05	183.84	181.88	181.80	2.17	181.88	181.80	12.93	165.59	173.27			
1.37	185.86	185.86	7.10	186.50	186.30	1.37	179.14	177.82	185.54	185.42	2.17	185.54	185.42	13.21	151.36	157.33			
2.17	185.13	185.13	7.10	186.42	186.39	1.37	184.05	184.72	186.54	186.44	2.17	186.54	186.44	14.98	164.17	172.86			
2.40	186.17	186.17	7.10	182.80	182.87	1.37	185.81	185.64	185.31	185.23	2.18	185.31	185.23	17.36	163.12	173.01			

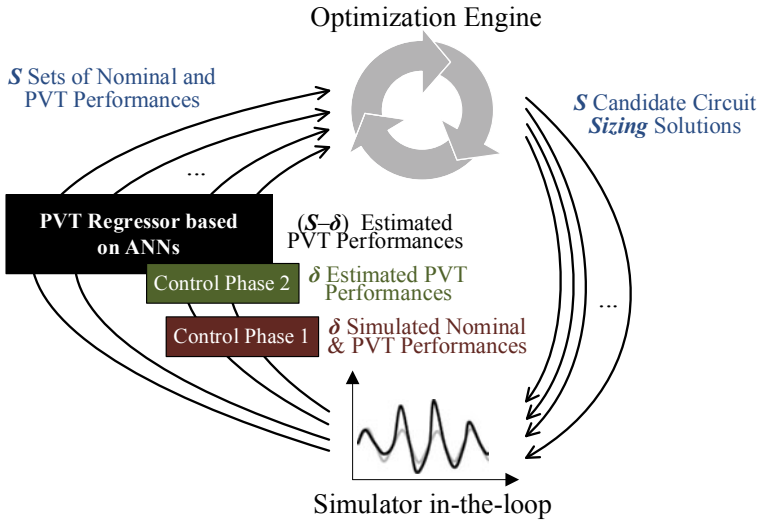


Fig. 4.7 Proposed controlled PVT regressor embedded into a simulation-based sizing loop

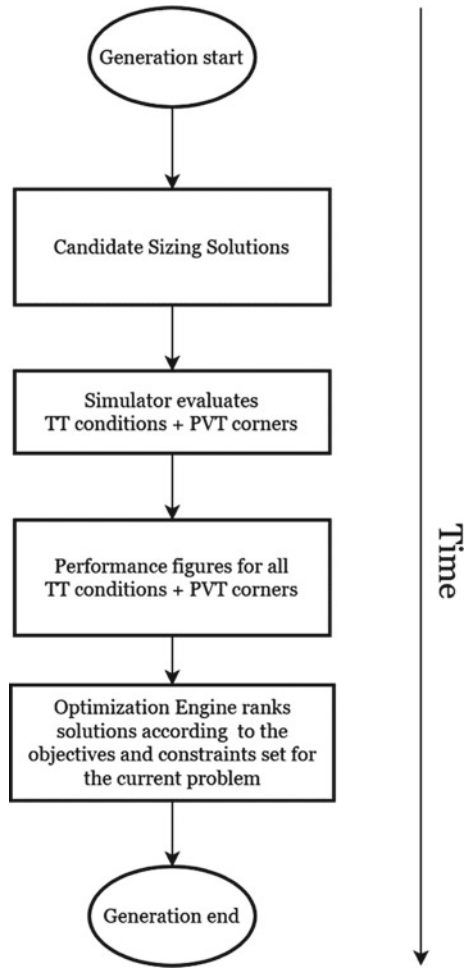
The simulator will only need to evaluate the solutions for TT conditions in the modified AIDA loop. Each of the ANNs will receive, as input, the sizing solution and, according to its tuning mode, the performance figures respective to the TT conditions (previously evaluated by the simulator). With the inputs defined, each ANN will output the performance figures corresponding to a specific corner and tuning mode, so the performance figures for all PVT corners can be sent to the optimization engine for further ranking. These performance figures are a mix of simulated performance figures (TT corners) and predicted performance figures (remaining PVT corners). In the scope of the sizing optimization, one of these loops represents a generation of the optimizer. A brief flowchart of one generation of the modified loop is shown in Fig. 4.9.

In the following sections, a comparison with an exhaustive circuit sizing optimization is performed to determine if the optimization with the modified loop, while using the PVT estimator throughout 100% of the optimization, achieves adequate results.

4.4.1 Class C/D VCO with PVT Estimator Working at 100%

This speed-up factor results from the optimization with the modified loop only evaluating the TT conditions for two tuning modes and the original optimization, which evaluated both TT conditions and PVT corners for 2 different tuning modes ($\frac{18}{2} = 9$). The number of generations of the now modified optimization was set to 350, the same as the original, to obtain the best comparison base possible. Both optimization

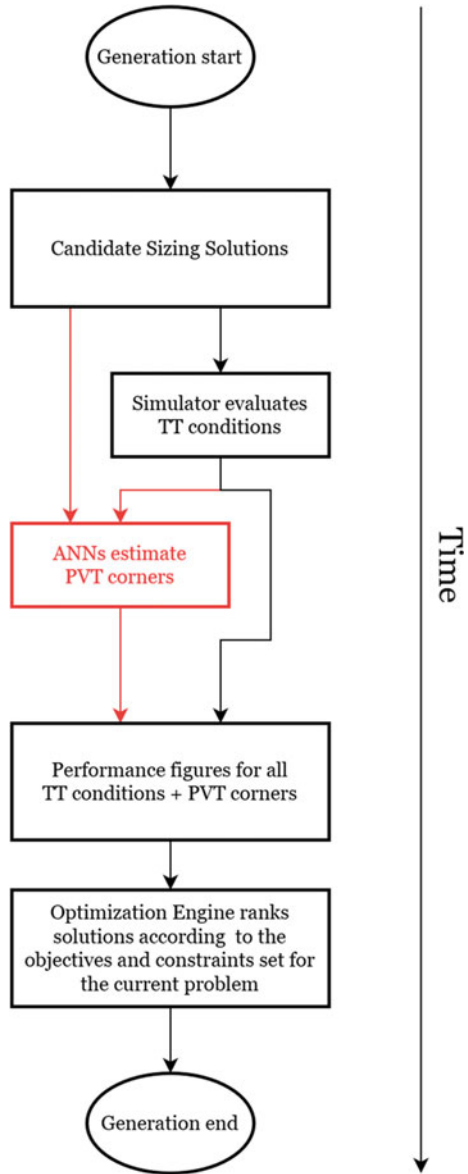
Fig. 4.8 Flow of a generation of the original AIDA loop



constraints and objectives were the same, shown in Table 2.5. The POF evolution throughout the modified loop optimization is depicted in Fig. 4.10 alongside the final original POF. However, to ascertain whether the solutions obtained are feasible or not, Unfortunately, all the solutions were revealed to be unfeasible in at least 30 of the 160 total performances. The POF values obtained in generation 350 and the number of failed optimization constraints are shown in Table 4.16.

As observable, the number of final solutions obtained increased substantially (from 27 to 40), and the range of values for both worst-case power and worst-case phase noise at 10 MHz improved significantly, with solutions where the worst-case power achieves 2 mW, less than half of the lowest worst-case power of the original optimization, and also, the worst-case phase noise of all solutions decreased by at least 2 dBc/Hz. However, these results cannot possibly be translated into a working

Fig. 4.9 Flow of a generation of the modified AIDA loop (modification in red)



circuit. Finally, a small optimization of 20 generations with the original AIDA loop was performed using the last state of the modified loop optimization as the starting point. This optimization was done to ascertain if feasible solutions could be found from this starting point. However, even after this step, no feasible solution was found.

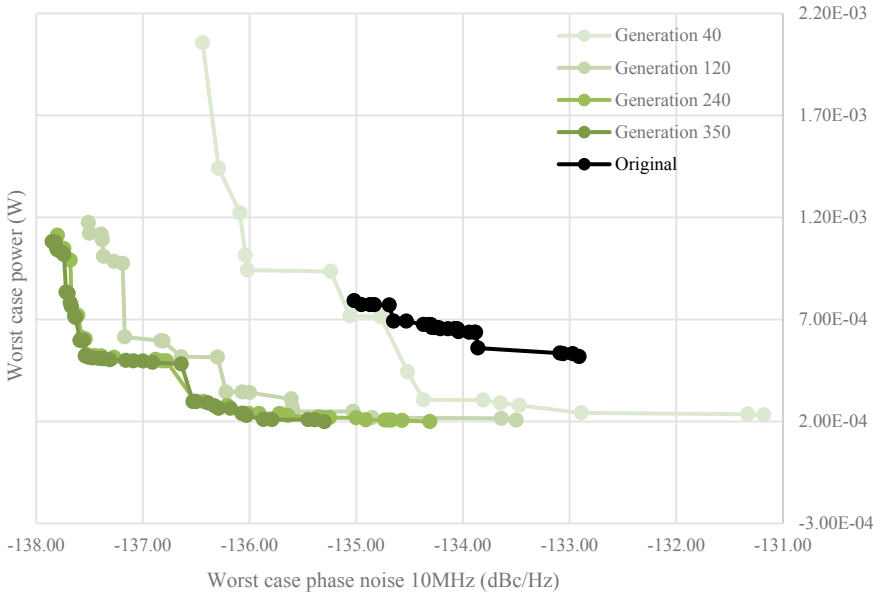


Fig. 4.10 POF evolution throughout the modified loop optimization

These results demonstrate one major flaw in the modified loop. It over-estimates the circuit performances, i.e., it makes the optimization engine search in a performance space where the circuit simply cannot operate in real-world conditions. In order to prevent this situation, some control has to be introduced in the modified loop to guide the optimization into feasible regions.

4.4.2 PVT Estimator with Error Controller

As stated, some control has to be introduced in the loop to guide the optimization to feasible solution regions better. In order to achieve this, a simple error controller for each ANN used in the modified loop was implemented. A brief flowchart of one generation of the modified loop with the new controller is shown in Fig. 4.11.

At each generation, before candidate sizing solutions and TT performance figures are sent to the ANNs, they first pass through a controller that will choose which ANNs will operate at the current generation, i.e., which PVT corner/tuning mode combination will be simulated or predicted.

First, the controller sends 20% of the candidate sizing solutions to be simulated and predicted at the same time. For the PVT corner/tuning mode combination, with the simulator's output, the controller checks if there are more feasible solutions than unfeasible solutions. If there are more unfeasible solutions than feasible solutions or the same number, the PVT corner/tuning mode combination will be simulated

Table 4.16 POF at generation 350

Worst case phase noise 10 MHz (dBc/Hz)	Worst case power	#Failed constraints	Worst case phase noise 10 MHz (dBc/Hz)	Worst case power	#Failed constraints
– 137.85	1.08E–03	30	– 137.31	5.02E–04	49
– 137.82	1.08E–03	31	– 137.16	4.99E–04	49
– 137.81	1.05E–03	36	– 137.09	4.96E–04	49
– 137.80	1.04E–03	35	– 137.00	4.96E–04	49
– 137.76	1.03E–03	36	– 136.91	4.89E–04	49
– 137.74	1.02E–03	42	– 136.64	4.82E–04	49
– 137.72	8.34E–04	36	– 136.53	2.97E–04	51
– 137.70	8.28E–04	36	– 136.50	2.97E–04	50
– 137.68	7.82E–04	35	– 136.39	2.91E–04	49
– 137.67	7.65E–04	36	– 136.33	2.77E–04	49
– 137.64	7.17E–04	36	– 136.29	2.64E–04	54
– 137.63	7.09E–04	41	– 136.18	2.64E–04	48
– 137.59	5.97E–04	44	– 136.07	2.39E–04	51
– 137.56	5.97E–04	36	– 136.06	2.39E–04	50
– 137.54	5.21E–04	49	– 136.03	2.29E–04	48
– 137.53	5.21E–04	49	– 135.87	2.10E–04	49
– 137.51	5.14E–04	49	– 135.79	2.09E–04	51
– 137.47	5.10E–04	49	– 135.45	2.09E–04	49
– 137.41	5.08E–04	49	– 135.39	2.08E–04	55
– 137.37	5.08E–04	49	– 135.30	1.99E–04	55

(instead of predicted) in that generation for the remaining candidate solutions. This acts as the primary filter to prevent the optimization of entering unfeasible regions.

If there are more feasible solutions than unfeasible, it passes to the next step. Here the controller uses the output of the feasible solutions from the simulator, i.e., simulated performances (from the previous step), and compares them to the corresponding predicted performances. The error between each performance is calculated and the average error of all points is obtained. If the error (MAPE) is higher than 5%, the combination will be simulated in that generation for the rest of the candidate solutions. If it is equal to or lower than 5%, the corresponding ANN will predict the PVT corner/tuning mode combination in that generation for the rest of the candidate solutions. The flow of the controller is depicted in Fig. 4.12.

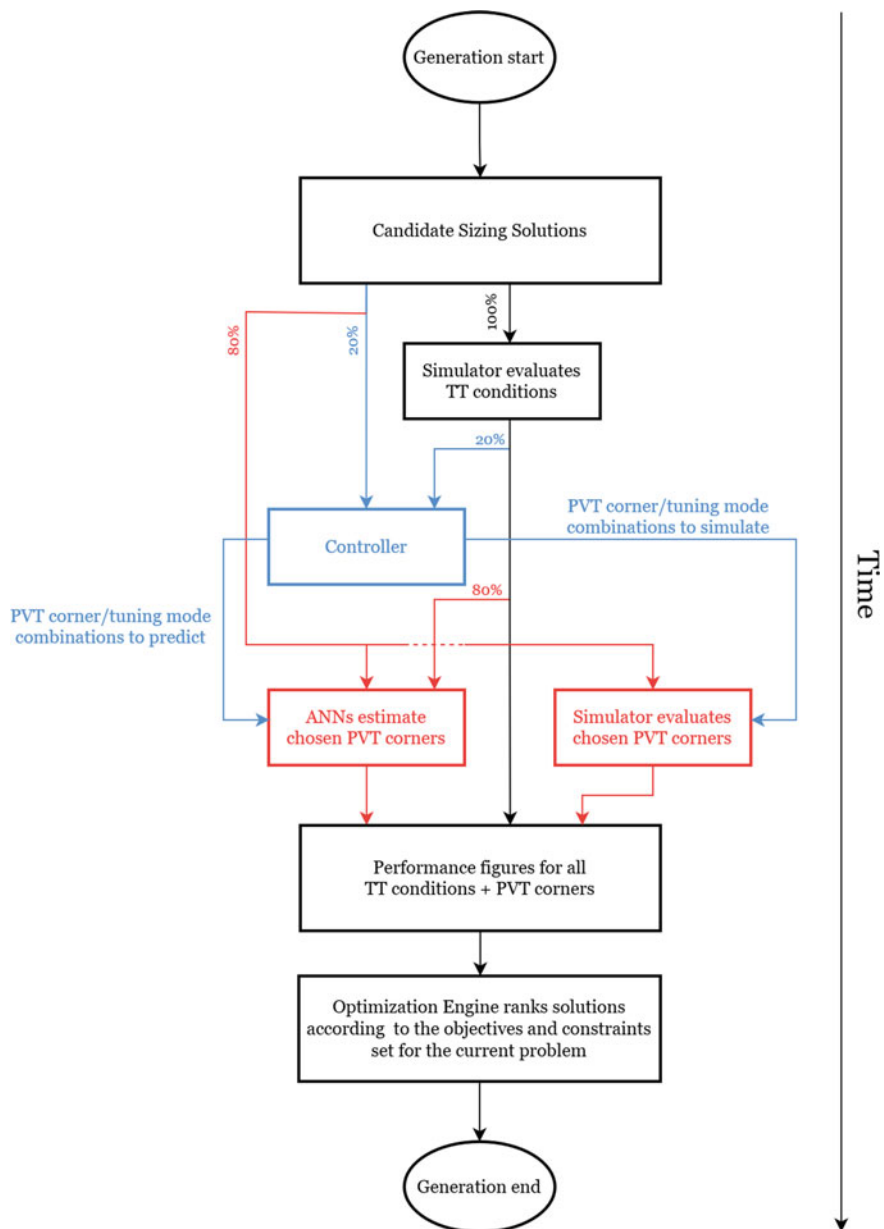


Fig. 4.11 Flow of a generation of the controlled modified AIDA loop (modification in red and controller in blue)

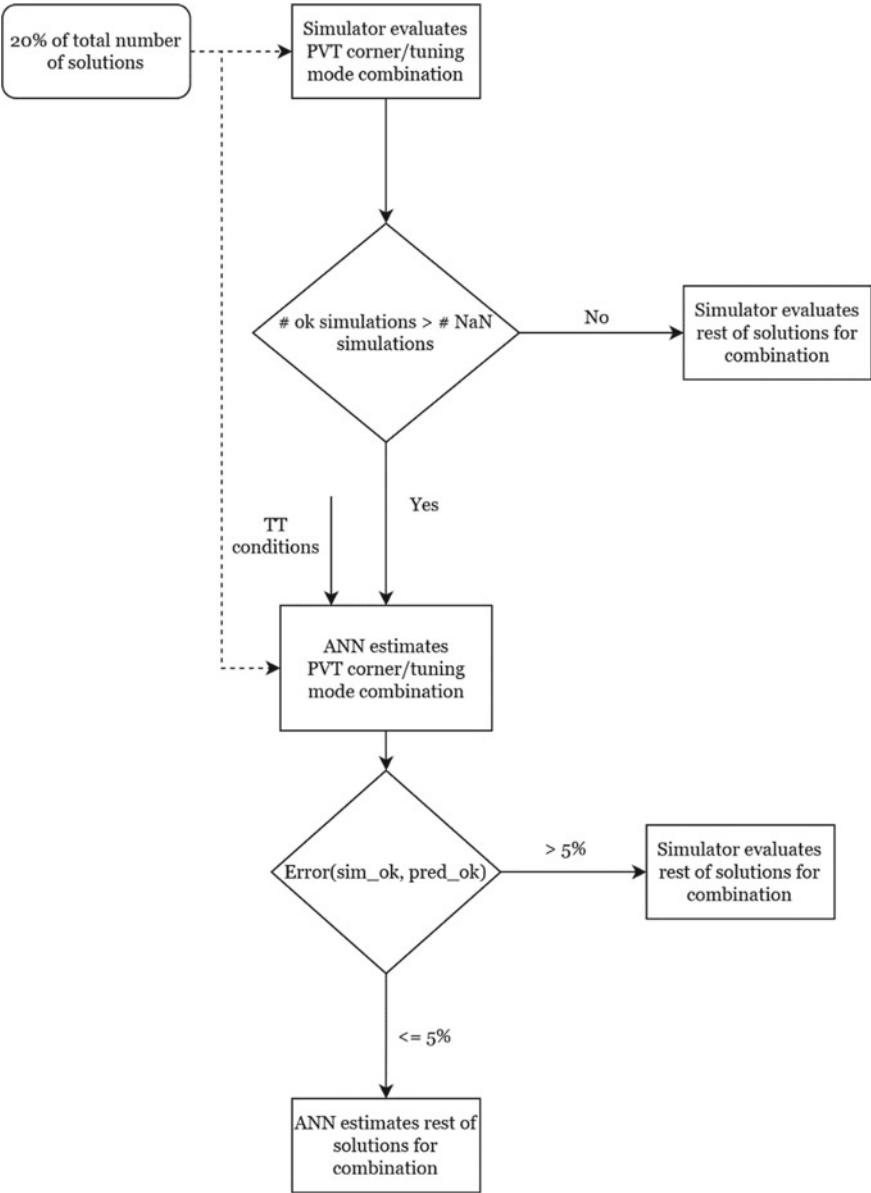


Fig. 4.12 Flow of the controller for each PVT corner/tuning mode combination

4.4.3 Results with Controlled PVT Estimator

Using the new controlled PVT estimator, three different optimizations will be performed, where in each one, a different objective of this dissertation will be analyzed:

- **Class C/D VCO for 3.5-to-4.8 GHz:** check if the controlled PVT estimator is capable of adequate circuit performance estimations in the PVT corners;
- **Class C/D VCO 2.3-to-2.5 GHz:** verify if the controlled PVT estimator can be directly reused for optimization with entirely different objectives and constraints of the same topology that was trained for, i.e., without re-training it (plug-and-play functionalities);
- **ULP Class B/C VCO:** verify if the same ANN structure used in the controlled PVT estimator for a particular circuit topology can be reused for a different VCO circuit topology (plug-and-train functionalities).

4.4.3.1 Class C/D VCO for 3.5-to-4.8 GHz

This section's main objective is to compare the final POF results using the controlled PVT estimator with the final POF obtained in the original optimization. The number of generations of the optimization using the controlled PVT estimator was set to 330, and in the last 20 generations, the optimization was carried out using the original loop. The POF evolution throughout the optimization with the controlled PVT estimator is depicted in Fig. 4.13 alongside the final original POF.

As observable, the optimization at generation 330, i.e., the optimization using the controlled PVT estimator, finds solutions with similar worst-case power and worst-case phase noise at 10 MHz to the original optimization when comparing with the optimization using a PVT estimator working at 100% in Fig. 4.10.

The circuit sizing solutions of the POF at generation 330 were simulated and the results show that 1 of the 20 solutions of the POF passed all the optimization constraints and thus, a feasible solution was found. Additionally, 2 solutions close to the optimization specifications were also found, with only 1 constraint failing. These results are shown in better detail in Table 4.17. This concludes that the controller could direct the optimization to feasible performance regions, which was the main problem of the optimization using the PVT estimator working at 100%.

After the last 20 generations, the optimization could find 30 solutions, and because these generations were performed entirely with the simulator, all of them are feasible. Comparing the performances of these solutions with the original POF, it is possible to observe that the solutions in terms of worst-case power are worse than the original POF by at least 1 mW, but regarding the worst-case phase noise at 10 MHz the optimization was capable of finding solutions with nearly less 2 dBc/Hz. The final POF is shown in Fig. 4.18 and the values are shown in detail in Table 4.18.

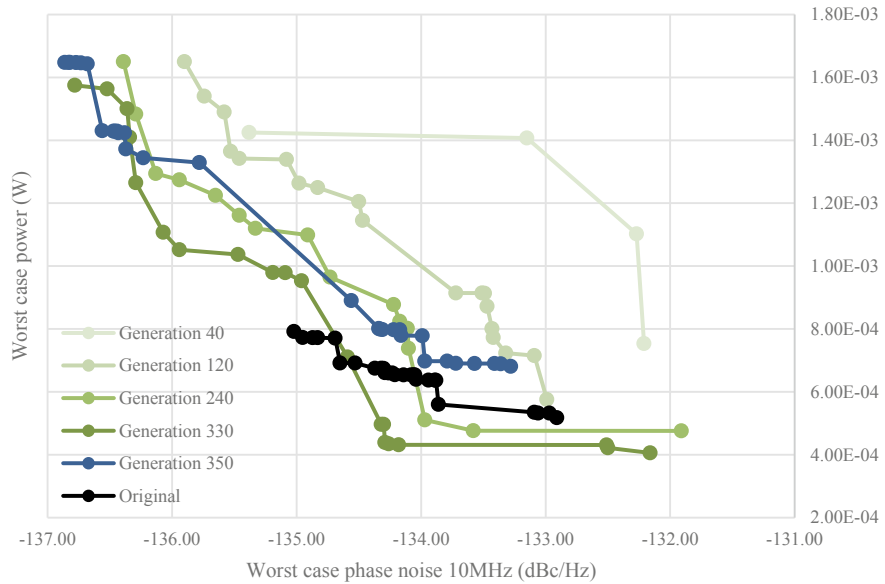


Fig. 4.13 POF evolution throughout the modified loop optimization with controller

Table 4.17 POF at generation 330

Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)	# Failed constraints	Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)	# Failed constraints
− 136.78	1.57	1	− 134.96	9.53×10^{-1}	24
− 136.52	1.56	13	− 134.59	7.11×10^{-1}	23
− 136.36	1.50	0	− 134.32	4.96×10^{-1}	30
− 136.34	1.41	1	− 134.30	4.96×10^{-1}	30
− 136.29	1.26	7	− 134.29	4.39×10^{-1}	30
− 136.07	1.11	13	− 134.26	4.35×10^{-1}	30
− 135.94	1.05	18	− 134.18	4.31×10^{-1}	30
− 135.47	1.04	13	− 132.51	4.31×10^{-1}	47
− 135.19	9.79×10^{-1}	18	− 132.50	4.22×10^{-1}	30
− 135.09	9.79×10^{-1}	13	− 132.16	4.06×10^{-1}	36

The speed-up obtained with the controlled PVT estimator is calculated by using the percentage of usage of each ANN throughout the first 330 generations of the optimization. These values are shown in Fig. 4.14.

While analyzing Fig. 4.14, it is possible to report that throughout the first 330 generations of the optimization, 78.5% of the original PVT corners evaluations were

Table 4.18 POF at generation 350

Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)	Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)	Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)
− 136.86	1.65	− 136.43	1.42	− 134.17	7.97×10^{-1}
− 136.83	1.65	− 136.38	1.42	− 134.16	7.79×10^{-1}
− 136.82	1.65	− 136.37	1.37	− 133.99	7.78×10^{-1}
− 136.77	1.65	− 136.23	1.34	− 133.97	6.98×10^{-1}
− 136.73	1.65	− 135.78	1.33	− 133.79	6.98×10^{-1}
− 136.68	1.64	− 134.56	8.90×10^{-1}	− 133.72	6.90×10^{-1}
− 136.56	1.43	− 134.34	8.01×10^{-1}	− 133.57	6.90×10^{-1}
− 136.47	1.43	− 134.33	7.99×10^{-1}	− 133.41	6.90×10^{-1}
− 136.46	1.43	− 134.31	7.98×10^{-1}	− 133.36	6.89×10^{-1}
− 136.44	1.43	− 134.22	7.98×10^{-1}	− 133.28	6.81×10^{-1}

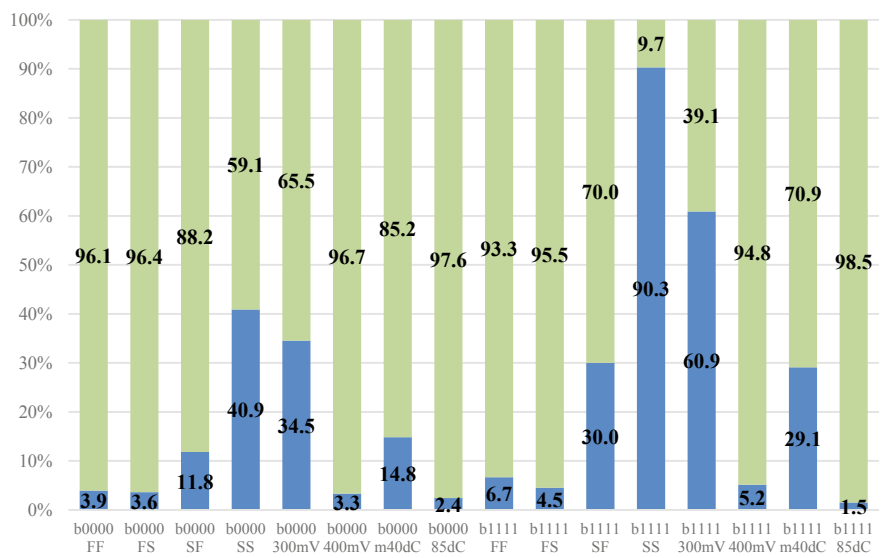


Fig. 4.14 Modes simulated (blue) versus predicted (green) throughout optimization

predicted using ANNs. Considering the simulation of the TT conditions, using the controlled PVT estimator resulted in a speed-up factor of 3.31.

However, the total speed-up of the 350 generations is lower due to the last 20 generations of full simulation that must be accounted. Considering these simulations, the total speed-up factor obtained using the controlled PVT estimator is 2.92.

These results show that the optimization using a controlled PVT estimator can find a small group of feasible or almost feasible solutions capable of competing in terms

of performance with the original optimization while achieving a speed-up factor of 3.31. In the original optimization, the first 330 generations took, approximately 577 h to complete, while the optimization using the PVT estimator only took 185 h to complete. Using the simulator for an additional 20 generations at the end of the optimization achieves a more robust and broad-ranging POF, while still achieving a speed-up factor of 2.92. With the PVT estimator, the modified optimization took 16 and a half days less than the original optimization. The main objective of this comparison is therefore confirmed to be possible, proving that the PVT estimator can find adequate PVT corner circuit performance estimations.

4.4.3.2 Plug-and-Play Class C/D VCO 2.3-to-2.5 GHz

The main objective of this section is to determine if the controlled PVT estimator, trained and tuned for the optimization of Sect. 4.4.3.1, can find adequate PVT corner circuit performance estimations for optimization with completely different targets of the same circuit topology. For the comparison, the optimization that will be used is a similar simulation-based sizing optimization of the circuit in Sect. 2.5. However, the range of the oscillation frequency in which the class C/D VCO will operate is now set to 2.3-to-2.5 GHz, and the optimization constraints were tightened, i.e., the maximum (or minimum) values chosen for the measured performances were decreased (or increased). The optimization constraints and objectives are shown in Table 4.19.

Table 4.19 Optimization constraints and objectives

Tuning mode	Measure	Testbenches	Units	Optimization constraint	Optimization objective
b_{0000}	f_{osc}	All	GHz	≥ 2.5	
	PN@10 kHz	All	dBc/Hz	≤ 54	
	PN@100 kHz	All	dBc/Hz	≤ 81	
	PN@1 MHz	All	dBc/Hz	≤ -103	
	PN@10 MHz	All	dBc/Hz	≤ -124	Minimize
	Power	All	mW	n/d	Minimize
	FOM@10 MHz	All	dBc/Hz	≥ 185	
b_{1111}	f_{osc}	All	GHz	≤ 2.3	
	PN@10 kHz	All	dBc/Hz	≤ -60	
	PN@100 kHz	All	dBc/Hz	≤ -87	
	PN@1 MHz	All	dBc/Hz	≤ -108	
	PN@10 MHz	All	dBc/Hz	≤ -129	Minimize
	Power	All	mW	n/d	Minimize
	FOM@10 MHz	All	dBc/Hz	≥ 185	

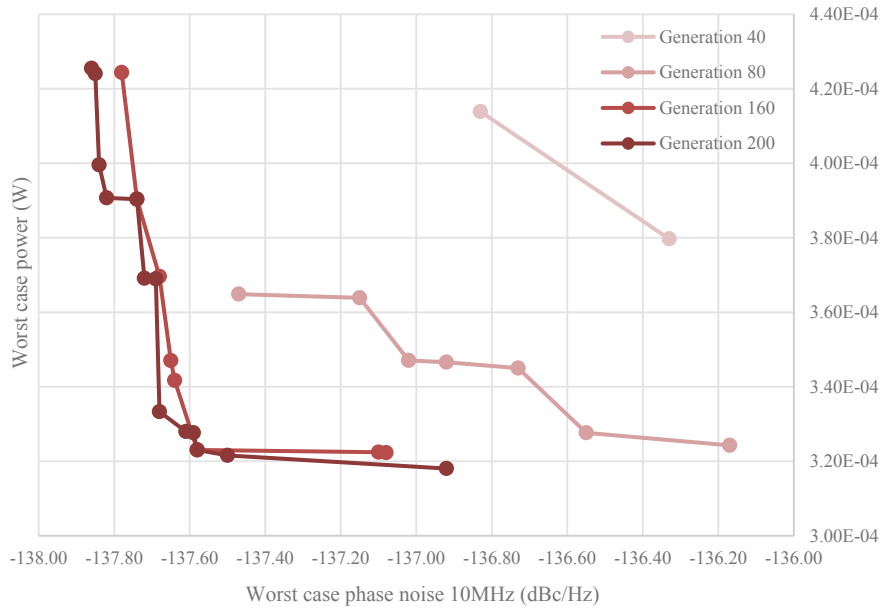


Fig. 4.15 POF evolution throughout the original optimization

First, an optimization using the unmodified loop was executed only for comparison terms. However, as opposed to the original optimization, the data will not be used for the training phase of the ANNs, given that it would defeat the main objective of this comparison. The optimization executed a total of 200 generations, took 350 h to complete, and resulted in 13 sizing solutions. Figure 4.15 shows the POF evolution of the original optimization, which contains the best sizing solutions throughout the generations and in Table 4.20 is shown the values of the final POF obtained at generation 200.

Similar to the optimization of the previous section, the optimization using the controlled PVT estimator was executed for 180 generations, and after this, the optimization was carried out for 20 generations using the original unmodified loop. The POF evolution throughout this optimization is depicted in Fig. 4.16 alongside the final original POF for easier comparison.

As can be seen by the results, the optimization at generation 180 was capable of finding solutions in a performance region with better worst-case phase noise at 10 MHz than the original optimization. However, the worst-case power almost doubled when comparing the solutions with the lowest worst-case power of both optimizations.

To ascertain whether the solutions found are feasible and how many optimization specification constraints were failed, if any, these were simulated. The results show that no solution passed all optimization specification constraints. However, 2 of them achieved only 2 failed constraints and 2 other achieved 7 failed constraints.

Table 4.20 Original POF at generation 200

Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)
− 137.86	4.25×10^{-1}
− 137.85	4.24×10^{-1}
− 137.84	4.00×10^{-1}
− 137.82	3.91×10^{-1}
− 137.74	3.90×10^{-1}
− 137.72	3.69×10^{-1}
− 137.69	3.69×10^{-1}
− 137.68	3.33×10^{-1}
− 137.61	3.28×10^{-1}
− 137.59	3.28×10^{-1}
− 137.58	3.23×10^{-1}
− 137.50	3.22×10^{-1}
− 136.92	3.18×10^{-1}

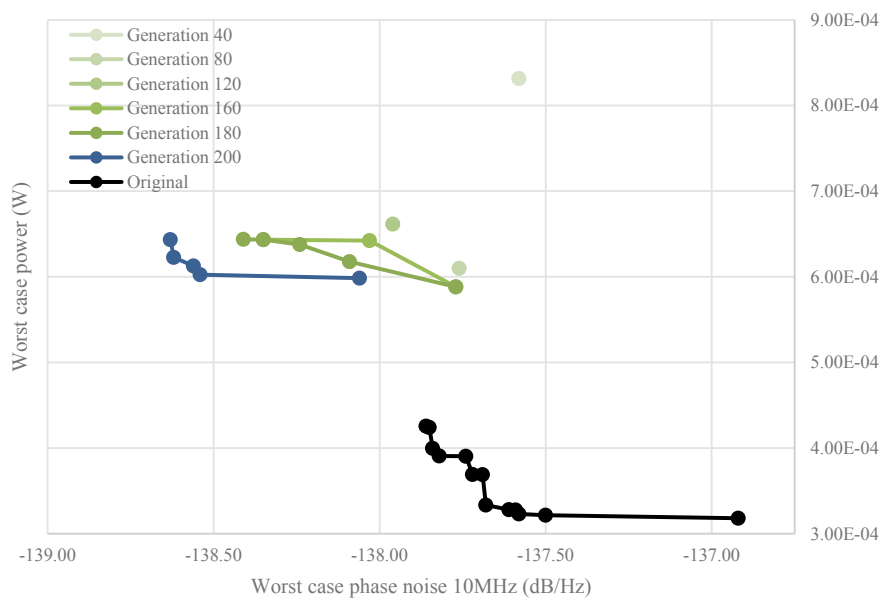


Fig. 4.16 POF evolution throughout the modified loop optimization with the controller

This means that these solutions came close to feasibility which indicates that the optimization was capable of finding an adequate POF performance region. These results are shown in Table 4.21.

After the last 20 generations, the optimization found 5 feasible solutions near the region of performances at generation 180. When comparing results, it is clear

Table 4.21 POF at generation 180

Worst case PN@ 10 MHz (dBc/Hz)	Worst case power (mW)	#Failed constraints
– 138.41	6.44×10^{-1}	7
– 138.35	6.43×10^{-1}	2
– 138.24	6.38×10^{-1}	2
– 138.09	6.18×10^{-1}	7
– 137.77	5.88×10^{-1}	12

that the lowest worst-case power solution found with the PVT estimator increased by almost 3 mW equaling a 100% increase from the original optimization. However, the optimization was capable of finding solutions with 1 dBc/Hz less than the solutions in the original POF. These results are shown in Table 4.22.

Once again, the speed-up obtained with the controlled PVT estimator is calculated by using each ANN's usage percentage throughout the optimization. These values are shown in Fig. 4.17.

While analyzing Fig. 4.21, it is possible to report that throughout the first 180 generations of the optimization, 74.5% of the original PVT corners evaluations were instead predicted using ANNs. Considering the simulation of the TT conditions, using the controlled PVT estimator resulted in a speed-up factor of 2.95.

However, as before, the total speed-up of the 200 generations is lower as the last 20 generations of full simulation must be accounted. Considering these simulations, the total speed-up factor obtained using the controlled PVT estimator is 2.48.

These results show that the controlled PVT estimator can find adequate circuit sizing performance regions despite being trained and tuned with data from another optimization, while still obtaining a speed-up factor of 2.95. In the original optimization, the first 180 generations took, approximately 315 h to complete, while the optimization using the PVT estimator only took 107 h to complete. Using the simulator for the additional 20 generations, resulted in a final POF with 5 feasible solutions while still being able to achieve a speed-up factor of 2.48. With the PVT estimator, in total, the modified optimization took 8 and a half days less than the original optimization. The final POF results reveal feasible solutions with better performances in terms of phase noise but worse performances in terms of power.

Table 4.22 POF at generation 200

Worst case phase noise 10 MHz (dBc/Hz)	Worst case power (mW)
– 138.63	6.43×10^{-1}
– 138.62	6.23×10^{-1}
– 138.56	6.13×10^{-1}
– 138.54	6.03×10^{-1}
– 138.06	5.98×10^{-1}

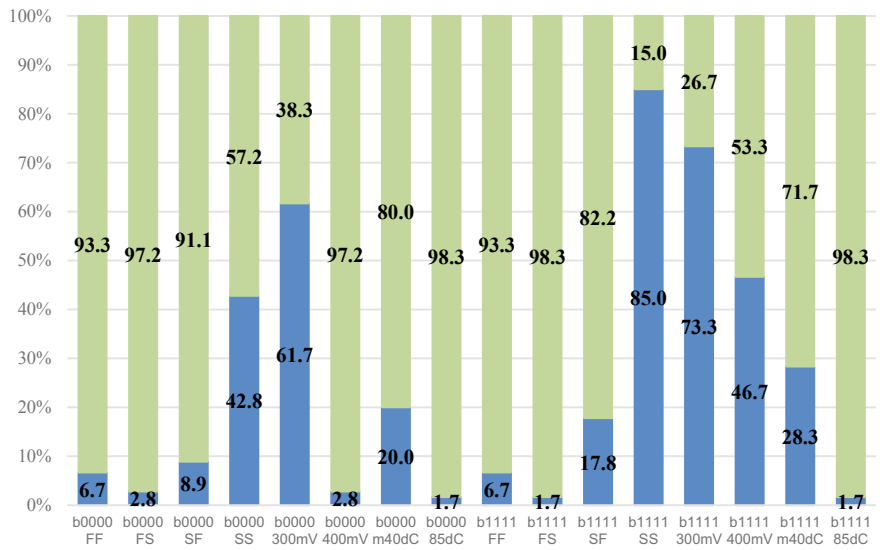


Fig. 4.17 Modes simulated (blue) versus predicted (green) throughout optimization

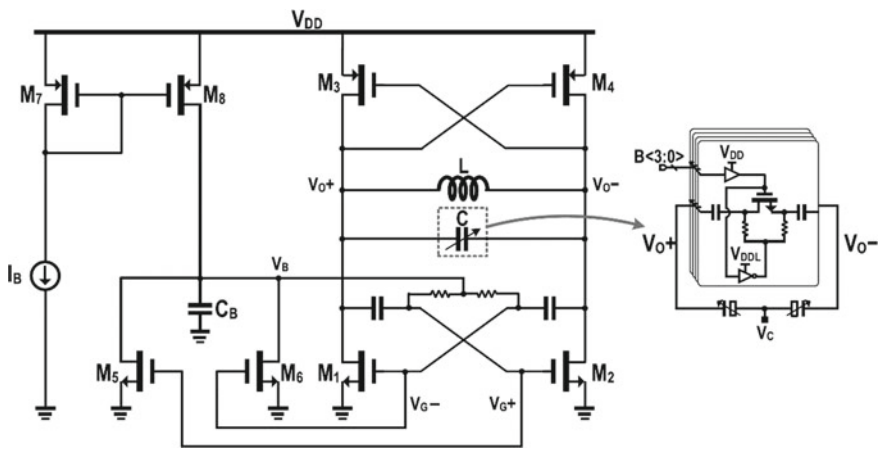


Fig. 4.18 Ultralow-power complementary class B/C hybrid-mode VCO topology [25]

Due to the random nature of the optimization loop, it is not certain that these performance results would be replicated if the optimization would be executed again. To ascertain the competitiveness of these solutions in terms of performances more optimizations would have to be performed. However, the reduction of optimization days using the PVT estimator reveals the competitiveness in terms of computational time used to obtain feasible solutions.

Table 4.23 Optimization variables

Variable	Units	Min	Grid	Max
irad	μm	15	5	90
itur	–	1	1	6
ispa	μm	2	1	4
iwid	μm	3	1	30
nccl, pccl	nm	60	20	240
nccf, pccf, $f^{5/6}$, $f^{7/8}$	–	1	1	32
nccw, pccw, $w^{5/6}$	μm	0.6	0.2	6
$l^{7/8}$	nm	130	20	6000
$w^{7/8}$	nm	120	20	6000
rccl	μm	0.8	0.2	30
cnv, cnh, vnv, vnh, snv	–	6	2	100
snh	–	6	2	50

4.4.3.3 Plug-and-Train Ultralow-Power Class B/C VCO

Next, we determine if the controlled PVT estimator, while reusing the ANN structure for a certain VCO circuit topology in the training phase, is capable of finding adequate PVT corner circuit performance estimations for a different VCO circuit topology. Simulation-based sizing optimization of the Ultra Low-Power class B/C VCO illustrated in Fig. 4.18 is used for this experiment. The complete list of optimization variables is presented in Table 4.23.

Similar to the optimization of Sect. 4.4.1, the principal objective was to minimize both power and phase noise at 10 MHz in both tuning modes, while imposing value constraints on 7 measured performances, in both tuning modes as well. These optimization constraints are shown in Table 4.24.

First, an optimization using the unmodified loop was executed for comparison with the optimization using the PVT estimator and to obtain the dataset to train the ANNs. The data from the original optimization was collected and structured similarly to the previous dataset. To use this dataset for the training phase of the ANNs, the pre-processing strategy was similar to the one described in Sect. 4.3, resulting in 16 datasets clear of outliers and null values, ready to be fed to the ANNs. The structure of the ANNs that will be used in the PVT estimator will be one tuned through this chapter, and used in Sects. 4.4.3.1 and 4.4.3.2. However, as observable in Table 4.23, the number of optimization variables is now 22, which forces the number of neurons in the input layer to change from 38 to 32 (22 optimization variables plus 10 performance figures in TT conditions).

The optimization with the unmodified loop executed a total of 100 generations, took 636 h to complete, and resulted in 14 sizing solutions. Figure 4.19 shows the POF evolution of this optimization and in Table 4.25 is shown the values of the final POF obtained at generation 100.

Table 4.24 Optimization constraints and objectives

Tuning mode	Measure	Testbenches	Units	Optimization constraint	Optimization objective
b_{0000}	f_{osc}	All	GHz	≥ 5.3	
	PN@10 kHz	All	dBc/Hz	≤ -55	
	PN@100 kHz	All	dBc/Hz	≤ -75	
	PN@1 MHz	All	dBc/Hz	≤ -95	
	PN@10 MHz	All	dBc/Hz	≤ -115	Minimize
	Power	All	mW	n/d	Minimize
	FOM@10 MHz	All	dBc/Hz	≥ 175	
b_{1111}	f_{osc}	All	GHz	≤ 4.6	
	PN@10 kHz	All	dBc/Hz	≤ -55	
	PN@100 kHz	All	dBc/Hz	≤ -75	
	PN@1 MHz	All	dBc/Hz	≤ -95	
	PN@10 MHz	All	dBc/Hz	≤ -115	Minimize
	Power	All	mW	n/d	Minimize
	FOM@10 MHz	All	dBc/Hz	≥ 175	

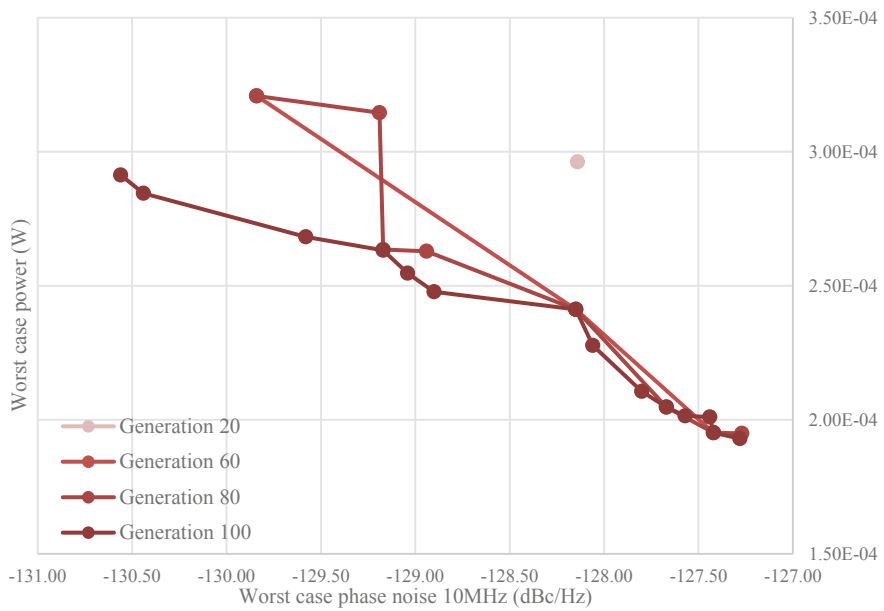


Fig. 4.19 POF evolution throughout the original optimization

Table 4.25 Original POF at generation 100

Worst case phase noise 10 MHz (dBc/Hz)	Worst case power (mW)
– 130.56	2.91×10^{-1}
– 130.44	2.85×10^{-1}
– 129.58	2.68×10^{-1}
– 129.17	2.63×10^{-1}
– 129.04	2.55×10^{-1}
– 128.90	2.48×10^{-1}
– 128.15	2.41×10^{-1}
– 128.06	2.28×10^{-1}
– 127.80	2.11×10^{-1}
– 127.67	2.05×10^{-1}
– 127.57	2.01×10^{-1}
– 127.44	2.01×10^{-1}
– 127.42	1.95×10^{-1}
– 127.28	1.93×10^{-1}

Using the same strategy as in the previous sections, the optimization using the controlled PVT estimator was executed for 80 generations, and after this, the optimization was carried out for 20 generations using the original unmodified loop. The POF evolution throughout this optimization is depicted in Fig. 4.20 alongside the final original POF for easier comparison.

As observable, the optimization at generation 80 could find 16 solutions in a performance region with considerably better worst-case phase noise at 10 MHz than the original optimization and significantly lower worst-case power throughout almost all solutions.

To ascertain whether the solutions found are feasible and how many optimization specification constraints failed, if any, these were simulated. The results show that no solution passed all optimization specification constraints. However, 5 of them achieved 6 failed constraints and 4 other achieved 12 failed constraints. These 9 solutions came close to feasibility which indicates that the optimization was capable of finding an adequate POF performance region. These results are shown in Table 4.26.

After the last 20 generations, the optimization found 12 feasible solutions near the region of performances at generation 80. It is evident that both worst-case phase noise at 10 MHz and worst-case power improved substantially when comparing with the original. The lowest and highest worst-case power decreased by 0.1 mW and 0.46 mW, respectively, while the lowest and highest worst-case phase noise at 10 MHz decreased by 1.8 dBc/Hz and 0.32 dBc/Hz, respectively. These results are shown in Table 4.27.

Once again, the speed-up obtained with the controlled PVT estimator is calculated by using the percentage of usage of each ANN throughout the optimization. These values are shown in Fig. 4.21.

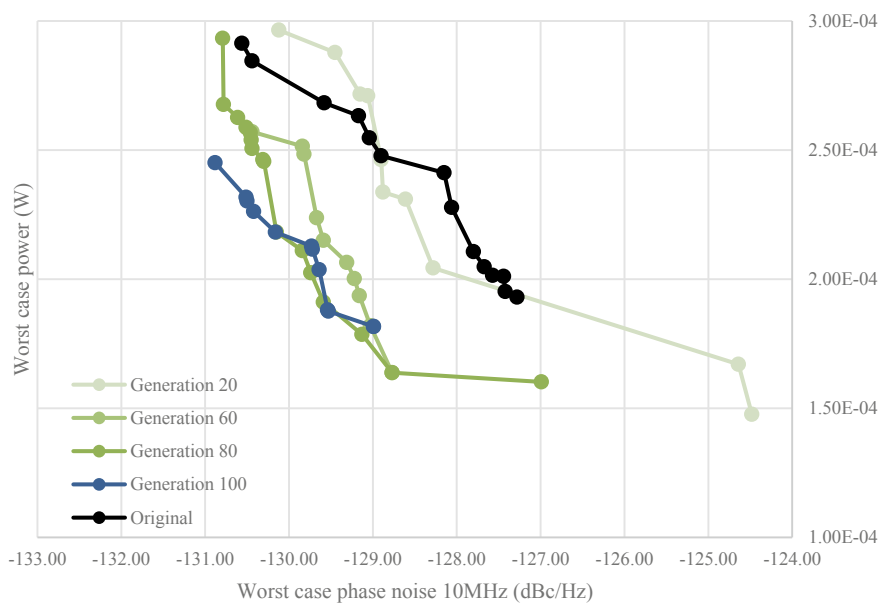


Fig. 4.20 POF evolution throughout the modified loop optimization with controller

Table 4.26 POF at generation 80

Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)	#Failed Constraints	Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)	#Failed constraints
− 130.79	2.93×10^{-1}	12	− 130.30	2.46×10^{-1}	24
− 130.78	2.68×10^{-1}	12	− 130.15	2.18×10^{-1}	6
− 130.61	2.63×10^{-1}	6	− 129.84	2.11×10^{-1}	6
− 130.51	2.59×10^{-1}	12	− 129.74	2.02×10^{-1}	6
− 130.46	2.57×10^{-1}	18	− 129.59	1.91×10^{-1}	30
− 130.45	2.54×10^{-1}	12	− 129.13	1.79×10^{-1}	6
− 130.44	2.51×10^{-1}	18	− 128.77	1.64×10^{-1}	24
− 130.31	2.46×10^{-1}	24	− 126.99	1.60×10^{-1}	42

When analyzing Fig. 4.21, it is possible to report that throughout the first 80 generations of the optimization, 74.1% of the original PVT corners evaluations were predicted using ANNs. Considering the simulation of the TT conditions, using the controlled PVT estimator resulted in a speed-up factor of 2.92.

However, the total speed-up of the 100 generations is lower because the last 20 generations of full simulation must be accounted for, and the percentage of simulation added of these generations is bigger when compared to the previous optimizations

Table 4.27 POF at generation 100

Worst case PN@10 MHz (dBc/Hz)	Worst case power (mW)
– 130.88	2.45×10^{-1}
– 130.51	2.32×10^{-1}
– 130.50	2.30×10^{-1}
– 130.42	2.26×10^{-1}
– 130.16	2.18×10^{-1}
– 129.73	2.13×10^{-1}
– 129.72	2.12×10^{-1}
– 129.64	2.04×10^{-1}
– 129.54	1.88×10^{-1}
– 129.53	1.88×10^{-1}
– 128.99	1.82×10^{-1}
– 128.99	1.82×10^{-1}

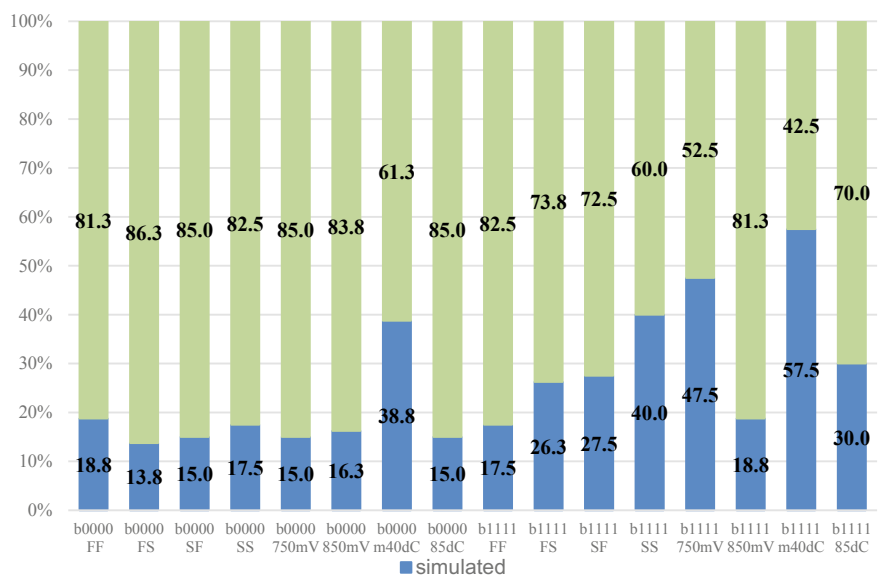


Fig. 4.21 Modes simulated (blue) versus predicted (green) throughout optimization

(20% of total optimization). Considering this, the total speed-up factor obtained using the controlled PVT estimator is 2.11.

These results confirm that the controlled PVT estimator can find adequate circuit sizing performance regions despite its ANNs not being tuned for this optimization, while still obtaining a speed-up factor of 2.92. In the original optimization, the first 80 generations took approximately 509 h to complete, while the optimization using the PVT estimator only took 174 h to complete. The additional 20 generations using

the original unmodified loop at the end of the optimization achieve better results than the original optimization while still achieving a speed-up factor of 2.11. With the PVT estimator, in total, the modified optimization took almost 14 days less than the original optimization. Once again, this reduction reveals the competitiveness of computational time used to obtain feasible solutions.

4.5 Conclusions and Future Research Directions

This chapter proposes an ML/DL method tailored for exhaustive PVT-inclusive RF IC sizing optimization problems for the first time in literature. Unlike previous approaches, where the simulator is entirely replaced, two control phases prevent the sizing loop from being guided to unrealistic design space regions. The proposed CPVTR was tested on a state-of-the-art class C/D VCO, providing speed-up factors of almost $3\times$ when compared with traditional PVT-inclusive optimizations, saving weeks of computational effort. Due to the characteristics of its structure, the same model can be used for optimizations with a different set of targets, i.e., generalizing beyond training.

4.5.1 Conclusions

This work presents an approach towards the acceleration of analog/RF IC optimization-based sizing loop with the help of a PVT corner performance estimator, using multiple ANNs, to complement the simulation process, therefore reducing the simulator workload.

For developing the PVT estimator, an optimization-based sizing of a Class C/D VCO for 3.9-to-4.8 GHz was used as a case study, gathering the necessary data to train the ANNs and ascertain if the results of the estimations before integration in the optimization loop, were adequate. All ANNs showed that the estimation error results were adequate, so the integration and discussion of the final results were performed.

Three different circuit optimizations were used to test the PVT estimator. The first one was the same optimization based-sizing of a class C/D VCO for 3.9-to-4.8 GHz used to develop the PVT estimator. The PVT estimator reduced 78.5% of the simulator workload, lowering the total optimization run time by 16 and a half days (the original run took 25 and a half days to complete). The final solution results showed similar performances to the original optimization, proving that the PVT estimator can find adequate PVT corner performances. The second optimization was an optimization-based sizing of the same circuit topology as the previous one, although the range in which the VCO operates was changed to 2.3-to-2.5 GHz, and the optimization constraints were tightened. The PVT estimator reduced 74.5% of the simulator workload, lowering the total optimization run time by 8 and a half days (the original run took 14 and a half days to complete). Feasible solutions were

found at the end of the optimization using the PVT estimator, proving its capability of being reused for optimizations with completely different targets of the same circuit topology its ANNs were trained to, demonstrating its plug-and-play functionalities. The third and final experiment was an optimization-based sizing of a different VCO circuit topology, i.e., an ultralow power class B/C VCO. The same structure of the ANNs used in the two previous tests was reused to train the ANNs in this optimization. The PVT estimator reduced 74.1% of the simulator workload, lowering the total optimization run time by 14 and a half days (the original run took 26 and a half days to complete). Feasible solutions with better performances than the original optimization were found at the end of the optimization using the PVT estimator, proving the capability of its ANNs reuse for a different VCO circuit topology, therefore demonstrating its plug-and-train functionalities.

4.5.2 Future Work

This work proved that an optimization loop using the PVT estimator could achieve feasible solutions while obtaining great speed-ups. However, it is possible to further optimize the competitiveness in both optimization times, quality of estimation, and the overall generalization of the PVT estimator when implemented in different circuit topologies.

While it was possible to prove that the modified loop can achieve feasible solutions, given the random nature of the optimization loop, each optimization can output different final solutions from another. Therefore, several optimizations must be performed to check if the final solutions are better or worse than the original optimization. As these optimizations would take much time, several weeks or months should be spent on this process.

The datasets used in this work were obtained by executing optimizations, with the unmodified loop, with an execution time of almost one month. Using larger datasets with a small number of *null* and duplicated values would surely increase the estimation capability of the ANNs, and given that smaller error results between estimation and real value would make the PVT estimator controller allow the ANNs to predict more often, the optimization would be faster, and the final solutions would be better.

An essential aspect of the optimization with the PVT estimator is its great execution time reduction. One hypothesis that was not performed due to time constraints was making the execution time of the optimizations with the PVT estimator the same as the original ones to verify if the final POF results would be better, given that in some instances, the number of generations would triple in number.

Augmentation of the capability of generalization of the PVT estimator could be done in two ways: using datasets with data from different circuit topology optimizations and making the PVT estimator online with the optimization. The first one would need a more complex structure for the ANNs used in the PVT estimator. However, there is a possibility that the ANNs would learn more information given different

circuit topology data. The second one is an approach that uses the ANNs so that a training phase would not be performed prior to the integration in the loop. Therefore, a dataset would not be needed. The ANNs would be integrated into the loop, and for the first generations, the loop would work without them. At each generation, the ANNs would use the data from the input and output of the simulator for training purposes, therefore learning the circuit topology. In each generation, the ANNs would estimate some candidate sizing solutions, using the actual values from the simulator to check the estimation error. When the error is low enough, the ANNs will become online on the modified loop at the start of the next generation, beginning the speed-up process only then. With this, the total speed-up obtained would decrease considerably, but a time-consuming optimization to gather the data for the dataset would not be needed.

References

1. Afacan E, Lourenço N, Martins R, Dündar G (2021) Review: machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test. *Integr VLSI* 77:113–130
2. Suissa A et al (2010) Empirical method based on neural networks for analog power modeling. *IEEE TCAD* 29(5):839–844
3. Wolfe G, Vemuri R (2003) Extraction and use of neural network models in automated synthesis of operational amplifiers. *IEEE TCAD* 22(2):198–212
4. Alpaydin G, Balkir S, Dündar G (2003) An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Trans Evol Comput* 7(3):240–252. <https://doi.org/10.1109/TEVC.2003.808914>
5. Liu H, Singhee A, Rutenbar RA, Carley LR (2002) Remembrance of circuits past: macromodeling by data mining in large analog design spaces. In: proceedings 2002 design automation conference, pp 437–442
6. Lourenço N et al (2019) Using polynomial regression and artificial neural networks for reusable analog IC sizing. In: 16th international conference on synthesis, modeling, analysis and simulation methods and applications to circuit design, pp 13–16, July 2019
7. Zhu K et al (2019) Genius route: a new analog routing paradigm using generative neural network guidance. In: proceedings of the ICCAD
8. Guerra D, Canelas A, Póvoa R, Horta N, Lourenço N, Martins R (2019) Artificial neural networks as an alternative for automatic analog IC placement. In: international conference on SMACD, Lausanne, Switzerland, July 2019
9. Gusmão A, Passos F, Póvoa R, Horta N, Lourenço N, Martins R (2020) Semi-supervised artificial neural networks towards analog IC placement recommender. In: IEEE international symposium on circuits and systems, Seville, Spain, Oct 2020
10. Gusmão A, Horta N, Lourenço N, Martins R (2022) Scalable and order invariant analog integrated circuit placement with attention-based graph-to-sequence deep models. In: expert systems with applications. Elsevier, Amsterdam
11. Gusmão A, Póvoa R, Horta N, Lourenço N, Martins R (2022) DeepPlacer: a custom integrated OpAmp placement tool using deep models. In: applied soft computing, vol 115. Elsevier, Amsterdam, 108188
12. Gusmão A, Horta N, Lourenço N, Martins R (2021) Late breaking results: attention in Graph2Seq neural networks towards push-button analog IC placement. In: ACM/IEEE design automation conference (DAC), San Francisco, USA, Dec 2021
13. Andraud M, Stratigopoulos H, Simeu E (2016) One-shot non-intrusive calibration against process variations for analog/RF circuits. *IEEE TCAS-I Reg Pap* 63(11):2022–2035

14. İslamoğlu G, Çakıcı TO, Afacan E, Dündar G (2019) Artificial neural network assisted analog IC sizing tool. In: 16th international conference on synthesis, modeling, analysis and simulation methods and applications to circuit design, pp 9–12, July 2019
15. Çakıcı TO, İslamoğlu G, Güzelhan ŞN, Afacan E, Dündar G (2020) Improving POF quality in multi objective optimization of analog ICs via deep learning. In: ECCTD, pp 1–4
16. Martins R et al (2019) Many-objective sizing optimization of a class-C/D VCO for ultralow-power IoT and ultralow phase-noise cellular applications. *IEEE Trans VLSI Syst* 27(1):69–82
17. Tensorflow. Accessed: Oct. 12, 2021. [Online]. Available: www.tensorflow.org
18. Keras. Accessed: Oct. 12, 2021. [Online]. Available: <https://github.com/fchollet/keras>
19. Aurlien Gron (2017) Hands-on machine learning with scikit-learn and tensorflow: concepts, tools, and techniques to build intelligent systems (1st edn). O'Reilly Media, Inc. ISBN 978-1491962299
20. Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. Cornell University. [arXiv:1505.00853](https://arxiv.org/abs/1505.00853), Nov 2015
21. Clevert D, Unterthiner T, Hochreiter S (2015) Fast and accurate deep network learning by exponential linear units (ELUs). Cornell University. [arXiv:1511.07289](https://arxiv.org/abs/1511.07289), Nov 2015
22. Early stopping with PyTorch to restrain your model from overfitting. Accessed: Oct. 12, 2021. [Online]. Available: <https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5>
23. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15:1929–1958
24. Passos F et al (2018) Enhanced systematic design of a voltage controlled oscillator using a two-step optimization methodology. *Integr VLSI* 63:351–361
25. Martins R, Lourenço N, Horta N, Zhong S, Yin J, Mak P-I, Martins RP (2020) Design of a 4.2–5.1 GHz ultralow-power complementary class-B/C hybrid-mode VCO in 65 nm CMOS fully supported by EDA tools. *IEEE Trans Circ Syst I Reg Pap (IEEE TCAS-I)* 67(11):3965–3977