TÉCNICO LISBOA

Accelerating Voltage-Controlled Oscillator Sizing Optimizations with a Convergence Classifier & Frequency Guess Predictor

João Luís Carreira Pich Domingues

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Professor Nuno Cavaco Gomes Horta

Doctor Ricardo Miguel Ferreira Martins

Examination Committee

President: Professor Tersa Maria de Sá Ferreira Vazão Vasques

Supervisor: Professor Nuno Cavaco Gomes Horta

Member of the Committee: Professor Rui Santos Tavares

November 2021

Declaration

Acknowledgments

I would like to acknowledge my supervisors, doctor Ricardo Martins and Prof. Nuno Horta, and also doctor Nuno Lourenço for all the support and help throughout the entire work of this Master Thesis. Their knowledge was extremely useful and valuable, being able to help answering some questions and some doubts that came along the way. I would also like to thank all the professors that I crossed with during my academic life, providing precious insights about every topic lectured, shaping the person that I am today.

I am grateful to all my colleagues and friends, whom I shared all these years, for their work and motivation that helped me go through this course. Their contribution for all the projects that we shared is incalculable.

I would like to give a final and warm words of gratitude and thanks for my parents and sister, for all their support and help so I could pursue my dreams. It is thanks to them that I managed to be where I am today, and to them I will always owe my success, thank you.

Finally, I would like to say thank you to Instituto de Telecomunicações for hosting me in its facilities. This research institution which is funded by FCT/MCTES through national funds and, when applicable co-funded EU funds under the project UIDB/50008/2020.

Abstract

The work presented in this thesis belongs to the scientific field of electronic design automation, with a special focus on the automatic sizing of radio-frequency integrated circuit blocks. With the help of deep learning and, more specifically, artificial neural networks, a new approach is introduced and discussed.

Machine and deep learning techniques are now being applied at different design levels of analog/radio frequency integrated circuits, from modelling to test, to learn any nonlinear input-output relationships. These new techniques are also challenging the conventional optimization-based sizing strategies, by creating models that map from the device's sizes to the specifications, or the opposite. This is done by replacing or complementing the time-consuming simulator in-the-loop.

The approach proposed in this work is based on a supervised learning scenario using artificial neural networks both for classification and regression. A convergence classifier will be used to predict if a certain simulation is likely to converge or not, and a frequency guess predictor to predict the oscillating frequency. This method will be implemented and tested on voltage-controlled oscillators' optimizations, by learning from a dataset of previous performances obtained by the simulator. The studied voltage-controlled oscillator circuit topologies are evaluated under extreme operation, i.e., Process, Voltage and Temperature corners. It is expected that these networks filter and discard solutions with no valuable information for the optimization loop, and thus, greatly reducing the overall time of the optimization process.

The result is a model that can predict non desired solutions, resulting in gains of almost 20% in overall simulation time, by discarding non valuable points. And it is able to correctly predict oscillatory frequencies, as the difference between these ones and the ones given by the simulator, reach values of mean absolute percentage error under 12%. Additionally, the use of this model does not compromise the results, as the ones obtained are very similar to the ones obtained without its use, and even better in some cases. The model demonstrates to be feasible for different optimization specifications, as well as for other examples of Voltage-Controlled Oscillators. The gains are similar, as the model is able to save 10% and 17%, respectively, and the results very promising, resulting in a model with a strong level of generalization.

Keywords: Artificial Neural Networks, Automatic Sizing Optimization, Electronic Design Automation, Radio-Frequency Integrated Circuits, Voltage-Controlled Oscillator.

Resumo

O trabalho apresentado neste relatório pertence ao ramo científico da automação de projeto eletrónico, com um foco especial na automação do dimensionamento de circuitos integrados de radiofrequência. Com a ajuda de técnicas de aprendizagem profunda, e mais especificamente, redes neuronais artificiais, é apresentada e discutida uma nova abordagem.

Diferentes técnicas de aprendizagem profunda estão agora a ser aplicadas a diferentes níveis do projeto de circuitos integrados analógicos e radiofrequência, desde a modelação ao teste, para aprender relações não lineares entre entradas e saídas. Estas novas técnicas desafiam também as estratégias convencionais de dimensionamento baseadas em otimização, criando modelos que mapeiam desde os tamanhos do dispositivo até às suas performances, ou o oposto. Isto é feito substituindo ou complementando o simulador presente no ciclo de otimização, cujo tempo de execução é geralmente elevado.

A abordagem proposta neste trabalho é baseada num processo de aprendizagem supervisionada, utilizando redes neuronais artificiais tanto para classificação como para regressão. Um classificador de convergência será usado para prever se um certo dimensionamento é provável que convirja durante a simulação, e uma rede de regressão estima a frequência para o qual esse dimensionamento deverá oscilar. Este método será implementado e testado numa ferramenta de otimização de osciladores controlados por tensão, analisando soluções anteriores cujas performances foram obtidas no simulador. Estas topologia de circuitos são avaliadas sob uma análise de condições extremas de funcionamento, sendo que as consideradas são temperatura, processo e tensão. Espera-se que estas redes filtrem e descartem soluções não viáveis, tendo um impacto significativo na redução do tempo total do ciclo de otimização.

O resultado é um modelo capaz de prever soluções não viáveis, resultando em reduções de cerca de 20% no tempo global de simulação. E é capaz de prever corretamente e frequências de oscilação, sendo que as diferenças entre estas e as obtidas pelo simulador atingem valores de erro percentual médio absoluto inferiores a 12%. Adicionalmente, o uso deste modelo não compromete os resultados, sendo os obtidos muito semelhantes aos obtidos sem a sua utilização, e até atingindo melhores resultados em alguns casos. O modelo também demonstra viabilidade para diferentes especificações de otimização, bem como para a sua aplicação quase direta em outros osciladores controlados por tensão. Os ganhos são semelhantes, sendo que o modelo consegue economizar 10% e 17%, respetivamente, e os resultados muito promissores. Resultando assim, num modelo com um forte nível de generalização.

Palavras-Chave: Automação de Projeto Eletrónico, Circuitos Integrados de Radio Frequência, Otimização Automática de Dimensionamento, Oscilador Controlado por Tensão, Redes Neuronais Artificiais.

Table of Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

AMS Analog Mixed Signal

ANN Artificial Neural Network

CAD Computer Aided Design

CMOS Complementary Metal-Oxide-Semiconductor

DNN Deep Neural Network

EDA Electronic Design Automation

IC Integrated Circuit

IoT Internet of Things

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

ML Machine Learning

mmWave Millimeter Wave

MOO Multi-Objective Optimization

MS Mixed-Signal

MSE Mean Squared Error

OTA Operational Transconductance Amplifier

POF Pareto Optimal Front

PVT Process Voltage Temperature

ReLU Rectified Linear Unit

RF Radio Frequency

SGD Stochastic Gradient Descent

SoC System On a Chip

SOO Single-Objective Optimization

SST Steady State

SVM Support Vector Machine

ULP Ultralow-Power

VCO Voltage-Controlled Oscillator

VLSI Very Large Scale Integration

Chapter 1

Introduction

This chapter presents an introduction to analog Integrated Circuit (IC) design with special attention to the challenges that come with the current optimization-based sizing methodologies. Besides, it is introduced the concept of Machine Learning (ML), and the possible use of Artificial Intelligence (AI) to automate analog IC sizing is analysed, more specifically analog and Radio Frequency (RF) devices or circuits.

1.1 Motivation

In the present day, the IC industry has, maybe 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, and security. Despite Moore's Law not really being observed anymore, the evolution of IC is still clearly observable every year, with designers building systems that are increasingly more complex, power-efficient, and integrated. These systems often have a combination of analog and digital sections, where most components are integrated into a single chip originating Mixed-Signal (MS) Systems on a Chip (SoCs).

Most ICs are implemented using digital or digital signal processing circuitry. However, analog circuits are the bridge between digital circuitry and physical devices with a nonstop increase in connectivity needs. Even though analog circuits only occupy a small fraction in MS SoCs, they require more effort to be built, as shown in Figure 1.1. Besides, the time to market and development costs make of electronic systems design a challenging task, being, therefore, fundamental to accelerate their design as much as possible.

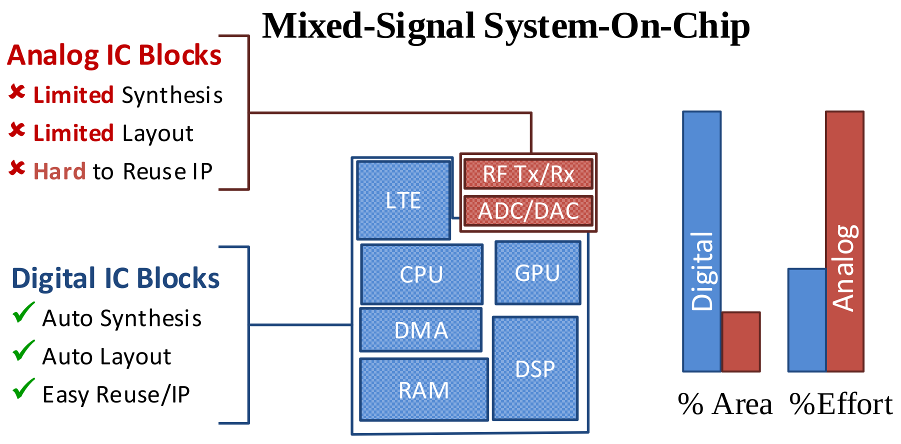

Figure 1.1 - Analog and Digital IC blocks. Reprinted from [1]

With Very Large Scale Integration (VLSI) technologies being broadly improved, it gave room for the growth of the global IC market, which was worth \$412.3 billion in 2019, is expected to grow to \$502.94 billion by 2023, growing at a compound annual growth rate of 5.09% [2], with analog/RF components being present in more than 50% of the total IC shipments yearly.

Electronic Design Automation (EDA) tools and design methodologies have been made available to cope with new capabilities offered by the integration technologies. However, there is still a huge discrepancy between the tools available for analog and digital IC design. 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, absolving the available resources. It is also easier to express a digital system, which can be represented naturally in terms of Boolean representation, 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 systems, they are responsible for design errors and expensive reruns. Therefore, economic pressure has motivated the pursuit of better methodologies to accelerate analog design.

The automation level for analog IC has been improving in the last years, being a field of profound academic and industrial research activity, which produces significant advances [4]. However, it is still far from the push-button stage, which leads to designers exploring the solution space almost manually as there are no standard advanced analog EDA tools and methodologies to automate the analog IC design flow.

On top of that, with predictions that more than half of new businesses will run on the Internet Of Things (IoT) and advances in telecommunications, such as the 5th generation broadband or 5G for short, there will be a huge demand for devices and sensors, opening doors to advances in areas such as healthcare, education, resource management, transportation, agriculture, and many other areas. Not only that, but there has also been an increase in the amount of data that is being continuously generated, resulting in new challenges within every part of the networks. Consequently, there is high pressure in today's market for large communication rates, extensive bandwidths, and ultralow-power consumptions. This is where RF ICs come in hand, playing a crucial role. This demand stresses out the problem which resides in the remarkable difficulty of RF and Millimeter Wave (mmWave) IC design in deep nanometric integration technologies for both IoT and 5G, due to their high complexity and demanding performances. Aggravated by the need to fulfill these at minimal costs and under frightful time-to-market constraints. Some of the design difficulties lie in the exceptionally wide range of frequencies and dynamic ranges involved, but also:

- 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 that are bearing variability issues and non-idealities which have never been experienced in older technology nodes.

One major objective for the design of modern RF ICs is to avoid the costs of redesign cycles, to diminish the post-fabrication tuning, and pursue first-pass fabrication success. Until now, the circuit designers were able to carry this flow manually thanks to the vast Computer Aided Design (CAD) tools provided by companies, but this method is no longer manageable due to the number of complex interactions and the sub-optimal RF designs that come with it.

1.2 Analog Circuit Sizing Flow

Over the past few years, the entire IC implementation from digital flow to graphic design system design flows, where one uses primarily stand-alone synthesis, placement, and routing algorithms to IC construction and analysis flows for design closure, has gone through significant changes. When it comes to analog design flow, most of the time when designing an Analog Mixed Signal (AMS) IC manually, the designer still follows the steps introduced by Gielen and Rutenbar [5], illustrated in Figure 1.2. The methodology consists of a series of top-down design steps that are repeated from the system-level to the device-level, and bottom-up layout generation and verification.

With the adoption of a hierarchical top-down design methodology, it became possible to explore complex system architectures, leading to a better overall system optimization at a higher abstraction level, even before doing more complex implementations at the circuit or device-level. Within this design methodology, an effort is made to find problems earlier in the AMS design flow to increase the success odds, while also having less or no overall time-consuming redesign iterations [6]. However, the increase of impact of layout parasitics and process disparities with the advances in technologies pushes the need to have many iterations in real-world designs. The number of hierarchy levels in this design flow is dependent on how complex is the system being handled, and despite not having a generally accepted representation for the design of the architecture, there are usually two design paths:

- Top-down electrical synthesis path, which includes the selection of the topology, translation of the specifications (or circuit sizing at lowest levels), and design verification;
- Bottom-up physical synthesis path, which includes layout generation and extraction, followed by verification of the specifications.

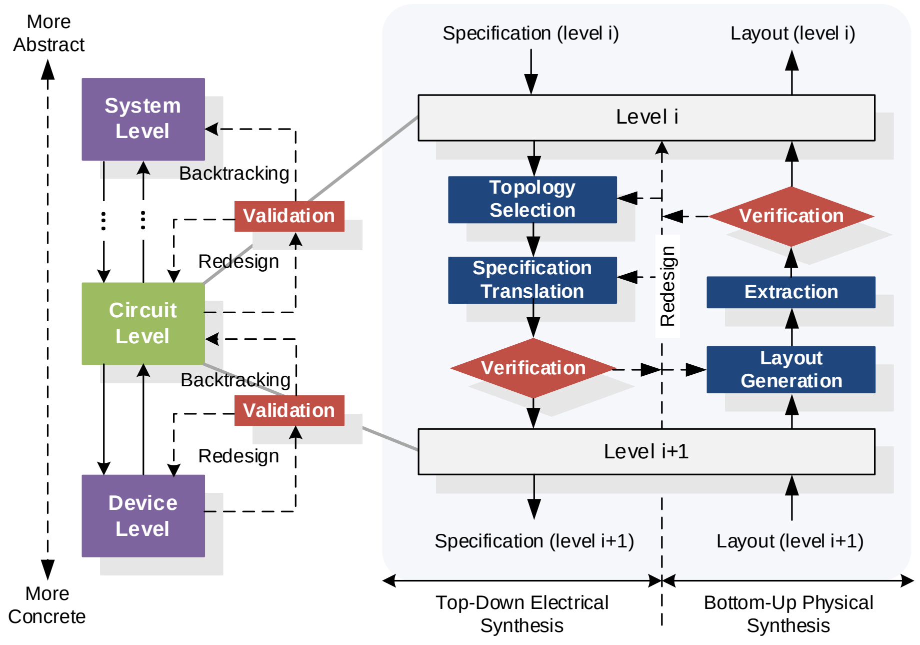

Figure 1.2 - Steps when designing AMS IC manually. Reprinted from [7]

Specifically, the step of determining the most suitable system or circuit topology is called Topology Selection. This one aim to meet the specifications given at the current hierarchy level, where the topology can be chosen from a set of available topologies or synthesized.

Specification Translation is the task encharged to map the high-level block specifications into independent specifications for each sub-blocks. This task culminates in-circuit sizing at the lowest level, where the sub-blocks are single devices. This stage is verified with the use of simulations. At higher levels in the design hierarchy, simulations are behavioural since no device-level sizing is accessible, wherein the lowest levels, device's sizes are available, so electrical simulations are used. Each block carries its specifications to the following level of the hierarchy, repeating this process until the top-down electrical synthesis flow is finished.

With advances not only in EDA tools for RF circuit designs but also in the development of RF process design kits, the productivity of RF engineers has seen major improvements. However, typical RF design is still influenced by the heavy work of design-tuning, where the design experience plays an especially important role [8]. Conventional analog IC design is particularly time-consuming due to the complicated nonlinear relationship between the design parameters and device/circuit/system specifications. The idea behind using ML techniques in analog/RF circuit design is to generate functional models of devices/circuits/systems that accurately mimic their functional behaviours and exploit them for different contexts.

The EDA community has been extremely helpful in overcoming some of these issues related to the design of RF IC, by offering optimization-based sizing methodologies. These tools use algorithms that automate the design space exploration, wherein industrial and academic levels the most accepted ones use a circuit simulator as the evaluation engine; in other words, simulation-based sizing. A scheme of a simulation-based sizing approach used to optimize the design of a RF IC is illustrated in Figure 1.3.

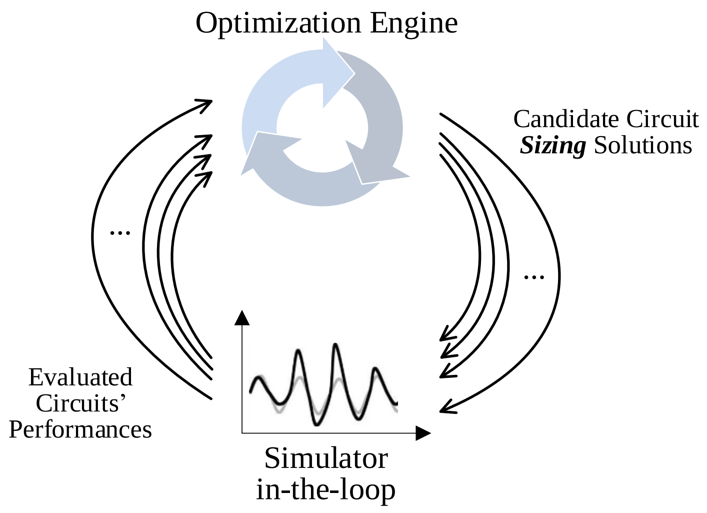
Optimization Engine
Candidate Circuit *Sizing* Solutions
...
...
Evaluated Circuits' Performances
Simulator in-the-loop

Figure 1.3 - Flow of a simulation-based RF IC sizing optimization

1.3 Objectives

This work has as a primary goal to accelerate the sizing optimization process of Voltage-Controlled Oscillators (VCOs) using ML techniques.

VCOs play an essential role in modern RF IC, being an open research and development topic. For instance, in IoT applications where the demand for Ultralow-Power (ULP) radios is significant, VCOs are extremely relevant design blocks [9]. 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, also need to 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 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.

Form the analysis of state-of-the-art sizing tools available for analog IC and RF design automation described in Chapter 2 of this dissertation, simulation-based sizing optimization tools are widely accepted approaches as they keep the solutions' accuracy high. Still, some challenges arise when designing complex VCO 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 of 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;
- As it is needed to provide a guessed oscillation frequency to the Steady State (SST), this one, is strongly correlated with the convergence analysis, promising designs may still be lost without simulating multiple guesses.

As described in [10], the optimization of a VCO using Mentor Graphics' ELDO [11] to simulate the circuit resulted in 59, 423 and 287 optimal sizing solutions for 3 different design targets, each taking about 100 hours in an Intel-Xeon-CPU E5-2630-v3@2.40 GHz with 64 GB of RAM using eight cores for parallel evaluation. Therefore, reducing the evaluation engine's workload, the simulator, which is the most time-consuming step in this optimization process, is extremely valuable.

Figure 1.3 depicts an optimization flow where each sizing solution can be simulated for extreme operations, suffering the challenges previously mentioned. The work will explore deep learning to enhance the optimization loop process two-fold. First, by building a classifier that predicts if the candidate solution's simulation is likely to converge, avoiding wasting computational time in non-convergent candidate solutions; and second, a regressor will predict the oscillating frequency guess that should be used in the simulation, improving the convergence rate of the simulator. It is desired that these models have high generalization capabilities, in order to be able to reuse them for different VCO circuit topologies. This aspect will be taken account in the development of these models.

1.4 Achievements

With the development of this work, it was produced a piece of software, in Python, that accomplishes the objective of identifying the valid configurations for simulation. This software trains, and produces ANNs capable of analysing a candidate circuit sizing solution, and identifying if it has value or not, and thus if it should be discarded or given to the simulator. Additionally, they are also able to guess the oscillatory frequency. The trained ANNs, are introduced and applied in an optimization flow, the AIDA framework, an analog integrated circuit design automation environment [12]. The ANNs are connected in a way that, it enhances this flow by receiving the sizing solutions and then sending the promising ones to the simulator, therefore, they reduce the workload of the simulator and support it on its convergence, thus the overall process time. Furthermore, these ANNs prove to be useful in, not only optimizing a specific circuit, but also for new constraints regarding the same one, and also for a different VCO, demonstrating a good level of generalization.

After having done a carefully analysis, the results obtained demonstrate to be very optimal, being able to accelerate the sizing optimization process of VCOs, by diminishing the workload of the simulator. On top of that, when comparing these results with the ones obtained without using the ANNs, the results are very similar, which show that the results were not compromised. The topmost achievement is, to bring this work to an even wider approach, where the ANNs would be able to be introduced in any VCO optimization, being its train done as the process occurs, hence saving additional time.

As a complementary of this thesis, is expected that a public scientific paper abording all the work here introduced and explained, be produced and published.

1.5 Organization

This report is structured as follows:

- Chapter 2 presents the state-of-the-art on analog IC sizing, presenting the work that exists nowadays, its advantages and disadvantages. After that, an overview of ML and some of its methodologies is made to estimate their applicability to analog IC automation;
- Chapter 3 presents the plan and methodologies of the proposed work, describing the development of the ANNs;
- Chapter 4 presents the study of the training process of the ANNs for two VCOs, presenting its results and analysis;
- Chapter 5 presents the results of the implementation of the ANNs in the design of the circuits, where the results are analysed and discussed;
- Chapter 6 presents the final remarks of the work and a brief discussion on future research directions.

Chapter 2

State-of-the-art

This chapter presents the state-of-the-art analog IC sizing automation methodologies. In the last two decades, the main tools used to automate the analog/RF circuit sizing are divided into two categories, knowledge-based and optimization-based. This classification is shown in Figure 2.1.

A brief introduction about ML is presented, how did it appear, why, and the demands that make its use so appealing. The methods to evaluate data will be discussed, their advantages and disadvantages, and the most appropriate one to use will also be presented.

2.1 Knowledge-based Sizing

For knowledge-based, tools like IDAC [13] and BLADES [14] have tried to systematize the design by making use of a design plan obtained from expert knowledge. These tools produce a pre-designed plan with the use of design equations and a design strategy that build component sizes that meet the performance requirements. This approach had good results to automatic analog IC sizing, being its main advantage of the short execution time. However, the process of deriving the design plan is complex and requires a lot of time. The unceasing supervision required to keep the design plan up to date with advances in technologies and the fact that the results obtained are not perfect makes this approach only suitable as a first-cut-design.

2.2 Optimization-based Sizing

Aiming for optimality, the next generation of sizing tools have been applying 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.

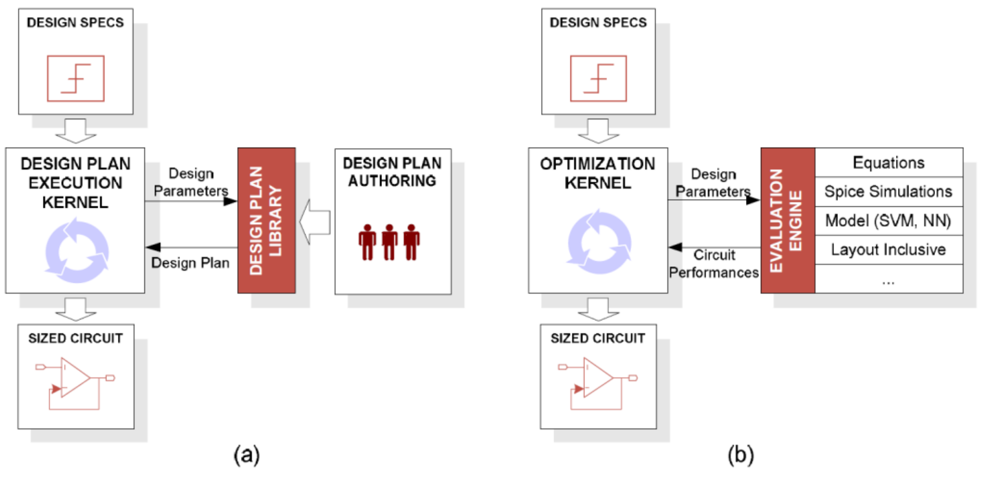

Figure 2.1 - Automatic circuit sizing approaches: (a) knowledge-based; (b) optimization-based. Reprinted from [15]

2.2.1 Equation-Based Evaluation

These methods use analytic design equations to describe the circuit performance. To resolve the degrees of freedom, tools such as OPASYN [16] and CADICS [17] were used, performing rule-based topology selection. However, the design equations still had to be deduced and ordered by hand, therefore, the symbolic simulator ISAAC [18] was developed to automatically produce, in a less complex way, design equations used to evaluate the circuit performance, while also reducing the introduction time for new circuit schematics. In [4], a synthesis experiment for a pulse detector frontend, which consists of a charge-sensitive amplifier and a 4-stage pulse-shaping amplifier, was conducted. Circuit synthesis can be described as the automatic process to determine the dimensions of the devices, such that the resultant circuit achieves the specifications of a given designer. The results are showcased in Table 2.1, showing a reduction in power consumption with a factor of 6.

Table 2.1 - Results of a synthesis experiment

Performance	Specification	Manual	Synthesis
peaking time	<1.5µs	1.1µs	1.1µs
counting rate	>200Khz	200Khz	294Khz
noise	< 1000rms e^-	750rms e^-	905rms e^-
gain	20V/fC	20V/fC	21V/fC
output range	> -1...1 V	-1...1 V	-1.5...1.5 V
power	minimal	40mW	7mW
area	minimal	0.7mm^2	0.6mm^2

The problem of using these methods is that the mapping of design characteristics by analytic equations is not straightforward and the approximations introduced in the equations result in poor accuracy.

2.2.2 Simulation-Based Evaluation

Most of the used tools have a simulation-based approach to evaluate the circuit's performance, retaining its main advantage of the generality and easy-and-accurate model. However, SPICE-based circuit synthesis's cost becomes too much time consuming due to the need to run many simulations to achieve the targeted performances.

In [12] this approach was addressed, where an analog IC circuit design automation environment, AIDA, is presented. This one implements a design flow from a circuit-level specification to a physical layout description, in which AIDA-C performs an automated circuit-level synthesis. AIDA-C is a tool with a multiobjective multi-constraint optimization approach that takes corners and the electrical simulator as an evaluation engine to address robust design requirements. 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, and temperature. AIDA was used to design several analog and RF circuits, where the results are compared and validated by industrial simulators and analysis tools such as ELDO and SPECTRE.

The AIDA-C architecture is based on a multiobjective evolutionary optimization kernel NSGA-II [19] using the simulator as an evaluation engine. The circuit test benches and the circuit itself are the designers' inputs, in the form of SPICE netlist(s), that have as parameters the optimization variables and provide the means to measure the circuit's performance. The designer is the one who defines ranges for the optimization variables, design constraints, and the goals of the optimization. The general flow of these techniques is shown in Figure 2.2.

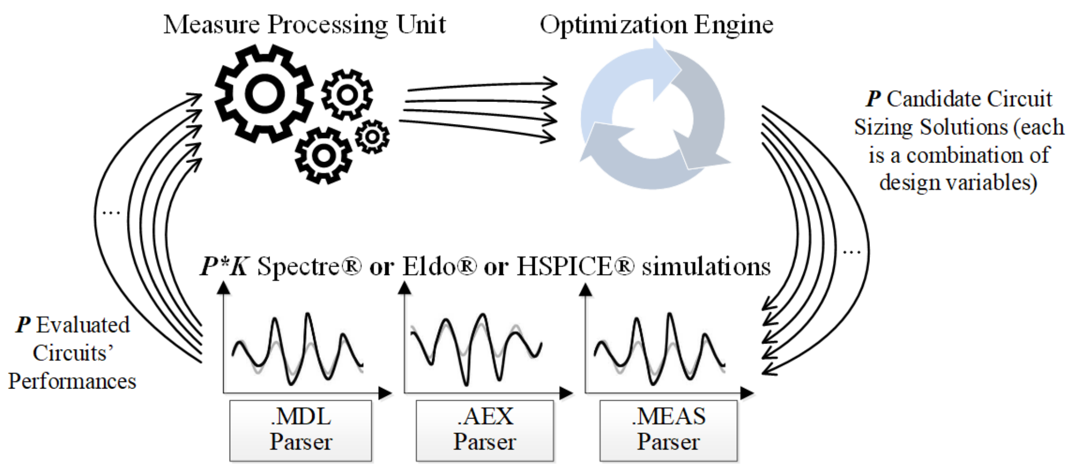
Measure Processing Unit
Optimization Engine
P Candidate Circuit Sizing Solutions (each is a combination of design variables)
P*K Spectre® or Eldo® or HSPICE® simulations
P Evaluated Circuits' Performances
.MDL Parser
.AEX Parser
.MEAS Parser

Figure 2.2 - Results of a synthesis experiment. Reprinted from [9]

The architecture represents the 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, wherein in each iteration, the framework simulates the several test benches, *K*, affected by each sizing of *P*, to extract the desired measures. There are several commercially available solutions to choose the simulator, for instance, Cadence's SPECTRE, Synopsys' ELDO, or Mentor Graphics' HSPICE. It is also possible to combine measures from different test benches into composed expressions, thanks to a measure-processing interface, which can be used as targets of the constrained multiobjective optimization problem.

Simulation-based evaluation has been used in a lot of actual works that aim to optimize RF circuits, being some of the most recent ones shown in Table 2.2.

Table 2.2 - Summary of Simulation-Based RF optimization

Reference	Method	Contribution
[20], 2012	HSPICE and CALIBRE	Solves the limitations of parasitic-included equivalent circuit models and predefined layout templates
[21], 2014	ELDO and HSPICE	Optimize both phase noise and power consumption
[22], 2016	HSPICE-RF	Layout-induced parasitic aware RF circuit sizing tool
[23], 2017	Electromagnetic simulations	Two-step design methodology
[24], 2018	HSPICE-RF	Parasitic-aware multi-objective RF circuit synthesis tool and performance space exploration
[25], 2019	HSPICE-RF	Exploits physical based parasitic circuit models for passive components

As each simulation made within an optimization-based loop is a time-consuming process, there has been a development in techniques to reduce the workload of the simulator. Particularly, Support Vector Machine (SVM) classifiers and ANN models have been extremely popular when addressing this problem.

2.3 Simulation-based Enhanced by ML

For many years, humans have been trying to develop machines to help them in daily tasks, to reduce their workload, and to achieve better results. As the advances in technologies kept increasing, the need to develop more complex and automated machines kept growing. This led to the birth of ML. ML is an area of AI that aims at building an expert system, focusing on the statistical properties of data.

The concept of ML was coined in 1959 by Arthur Samuel, having its foundations from probabilistic and statistical theorems such as the Probability Theory by Thomas Bayes. The first ANN was introduced in 1951 but only became relevant after Frank Rosenblatt's perceptron [26] and back-propagation [27], in 1958 and 1986, respectively.

One crucial characteristic of ML systems is their different categories, which depend on the amount and type of supervision. There are currently three different supervised levels, supervised learning, unsupervised learning, and semi-supervised learning.

Supervised Learning

For these models, the dataset used must have some observations and the expected results for those, called labels. These labels can further divide this method into two problem-solving techniques, classification, and regression.

A **classification** problem is when the output variable is part of a group, for instance, "dog" or "not dog", where **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, SVMs, and ANNs.

Unsupervised learning

For these models, the research use data that has not been labelled yet, aiming to create models that can draw inferences from datasets to describe hidden structures. The problems that his method targets can be further divided into clustering and association.

A **clustering** problem is characterized by the disclosure of the inherent groupings in the data, for instance, grouping people based on their age, where an association problem aims to discover relations rules within the data, such as people which buy X also tend to buy Y. Some examples of relevant algorithms are principal component analysis, K-means, and mixture models.

Semi-supervised learning

For most of the data like image processing and text processing, there is an abundant supply of unlabelled data, making it necessary for human experts to label this data manually, being this a very hard-working task. Semi-supervised learning appears as a more recent approach and is halfway between supervised and unsupervised learning making use of both labelled data and unlabelled data. Some examples of relevant algorithms are deep belief networks and autoencoders.

There have been made advances in technologies, which made it possible to have another type of optimization-based sizing tool, the numerical-model-based. These tools use macro models, for example, ANN or SVM, to accelerate the evaluation of the circuit's performance, reducing the high execution times caused by using only the electrical simulation inside the optimization loop, especially at the system-level.

2.3.1 Enhanced by Support Vector Machine

SVM is a supervised learning algorithm for data separation, that make use of linear combinations to produce a boundary that maximizes the margin between classes. This algorithm is especially good 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 the work presented in [15], a SVM classifier is used in a multiobjective case to enhance the robustness of the solution. An analog IC sizing tool, GENOM-POF, was used to illustrate the methodology, and to support the effect of corner cases on the Pareto Optimal Front (POF). The SVM is used to create feasibility models that diminish the design search space during the optimization process, hence reducing the number of required evaluations. This approach was validated by 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 fractional 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 on the number of required evaluations and larger gains in terms of time consumption. As the electrical simulation is more time consuming than the SVM model evaluation, it allows an efficient diminishing of the design search space.

In the work presented in [28], an SVM is also used to identify the feasibility design space of analog circuits to reduce a large amount of the entire design space, sampling only the points considered to be feasible and their neighbours. After choosing the right parameters of SVM's, the resulting model can have 100% accuracy on the training data. So, the difficulty relies on the generalization ability when facing an independent validation data set. To tackle this issue, three accuracy metrics were presented: Overall accuracy, percentage of false negatives, and percentage of false positives. These metrics are presented in equations 2.1, 2.2, and 2.3, respectively.

$$P_t = \frac{Number\ of\ correctly\ classified\ samples}{Number\ of\ samples\ in\ the\ validation\ set} \tag{2.1}$$

$$P_{fn} = \frac{Number\ of\ false\ negatives}{Number\ of\ positives\ in\ the\ validation\ set} \tag{2.2}$$

$$P_{fp} = \frac{Number\ of\ false\ positives}{Number\ of\ predicted\ positives} \tag{2.3}$$

To validate this method, two circuits were used as a case study, an Operational Transconductance Amplifier (OTA), and a low-voltage double balanced mixer. With the discard of the predicted negative values, the coverage of the feasibility design space by the committee, and it was always above 99% for both circuits, and the rate of feasible designs that the committee 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% less of that used by previous approaches.

The problem of using SVMs is that the tunning of hyper-parameters and the selection of the right kernel is quite difficult. SVMs also have a poor performance when faced with big datasets and a high number of features.

2.3.2 Enhanced by Artificial Neural Networks

ANN, or simply called neural network, is an algorithm based on the way the human brain analyses and processes information. ANN consist of node layers, one input layer, one output layer, and at least one, hidden layer. Each layer connects to the next one through links with its associated weight and threshold. ANNs can build effective end-to-end ML systems, being an exceptionally flexible construct. Learning methods have been derived from ANNs such as Deep learning, which is extremely useful when a lot of data is available.

In the work [29], presented in 2003, a neural network-based methodology is used to create fast and efficient models for estimating the performance parameters of Complementary Metal-Oxide-Semiconductor (CMOS) operational amplifier topologies. The results of both efficiency and accuracy of the obtained performance models were demonstrated in a generic algorithm-based circuit synthesis system. This tool is based on performance constraints that are defined by the user and aim to optimize a fitness function. The validation of the performance parameters of the synthesized circuits is done with SPICE simulations and then compared to the ones predicted by the neural network models.

The data used to train the models was generated directly from SPICE, therefore being able to provide values of accuracy like the ones obtained by SPICE simulations The ANN were categorized by three layers, one input layer, one output layer, and one hidden layer. To obtain the best generalization and accuracy results, the number of hidden layer neurons was alternated between 8 and 14. As activation functions, the hyperbolic tangent sigmoid function and the linear function was used for all hidden layer nodes and the output layer, respectively.

The ANN models showed to be much faster at collecting the data when comparing with having to collect data directly from SPICE, having a speedup factor of around 40,000 times, and the models proved to be able to capture nonlinear behaviour, with no need to require any knowledge or equations describing the internal structure of the op-amp, and this approach also proved to be generic and extensible, having the possibility to be used for other op-amp topologies.

In other works, some methodologies have been trying to diminish this issue by replacing this simulator with ANNs.

In the work presented in [30], it is used DNNs to replace SPICE. In addition to a Multiobjective Optimization (MOO), which is commonly used in analog circuits to identify the tradeoffs imposed by the designer specifications by using POFs, is used a Single-Objective Optimization (SOO). The ANN was trained using data obtained in the MOO phase, therefore requiring no additional step for its training. The ANN replaces the simulator in the later phase, in the SOO phase, reducing the performance evaluation time.

To verify this method, a two-stage amplifier and a folded cascode OTA circuit was used, having its results showcased in Table 2.3.

Table 2.3 - Results for two-stage amplifier and folded cascode OTA

Time			Accuracy				
SPICE [min]	Nets [min]	Improvement [%]		Gain	Bandwidth	Phase Margin	Power
800.7	27.07	96.62	μ_{error} (%)	0.00466	0.00502	0.00721	0.0456
			σ_{error}(%)	0.0173	0.01064	0.06115	0.10109

Time			Accuracy				
SPICE [min]	Nets [min]	Improvement [%]		Gain	Bandwidth	Phase Margin	Power
646.71	23.35	96.39	μerror (%)	0.01362	0.00437	0.08254	0.00716
			σerror(%)	0.10612	0.04452	0.18208	0.05252

Even though, the results obtained by avoiding time-consuming circuit simulations show that the evaluation speed increases substantially, there is a loss in accuracy that can only be recovered by resorting to the simulator in later stages of the optimization [31][32]. Additionally, these models are trained over the entire design space, spending relevant resources, and making use of combinations that would not be evaluated otherwise.

In another work [33], an ANN is also used to replace the simulator, however, to determine circuit performance instead of simulations, using generated data by preceding generations to train ANN that would otherwise be discarded. The main advantages that come with this work, is that there is no need for a data acquisition step to train ANNs, as these are trained within a classical simulation-based optimization tool, making it possible to use this approach for every new topology without loss of generality for all analog circuits. A Single Stage Amplifier and a Folded Cascode OTA circuit were used to validate this method. The results showed that the errors had values below 1%, and reductions in execution times by up to 64.8%, resulting in a 2.8x speed-up.

In [34] DNNs are used as a method to boost the optimizer's sample efficiency. With the use of an oracle, a comparison could be made between two designs, in terms of each design constraint, as a method to the selection of new designs. Since DNNs are especially good at approximating complex functions and have a good generalization to unseen samples, a DNN model is derived to imitate the conduct of such an oracle, which is in fact a simulator.

2.4 Other works on ML and Analog/RF Sizing

2.4.1 Predicting sizes from performances

The use of ANNs to find device sizing in analog IC is proving up to be a widely accepted approach and its use can learn and speculate circuit sizing when asked for some target specifications [35][36]. In [37] an ANN is developed to give the channel widths of all the transistors in a circuit when given the output specifications by the designer. The training phase data was performed with different SPICE parameters from the ones used in the test data to show the ability to give the transistor sizes of a circuit for new unknown 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 can estimate the current mirror circuits transistor sizes for never seen technologies, having a 94% accuracy. For the second one, a multilayer perceptron and the results had an accuracy of 90%.

In [38] 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 train and tune of such a model have proven to be exceedingly difficult. Having only used 277 handmade sizing solutions for the training phase, this one still needed an outer loop to acquire the model's hyperparameters, which reflected in a train of over 5 hours on such a short dataset. In [39] the sizing for an amplifier is also predicted using ANNs when given its specifications. However, in this one, the model and training phases are different since the test was only performed on 10 samples from the original dataset, and there is not made evaluation on the performance and usability of the model for unknown target specifications.

2.4.2 Reinforcement learning

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, getting rewards or penalties according to what action it performs, as shown in Figure 2.3. 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.

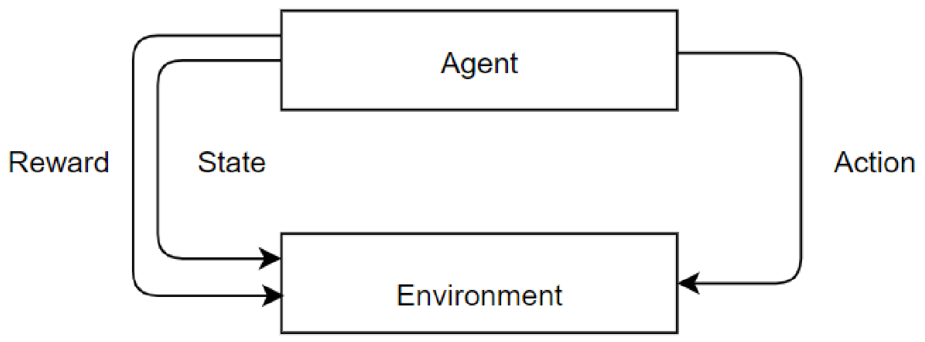

Figure 2.3 - Reinforcement learning interaction in agent-environment

This method has been used in alternating current optimization in [40], [41], and [42]. With the use of deep learning, an agent is trained, having no need for previous knowledge about optimizing circuits.

2.4.3 Other works

In the work [43], presented in 2019, a feed-forward time-delay ANN is used to address one of the biggest concerns in today’s SoC design and verification, power consumption. Analysing power consumption is still an extremely hard task because of the dependency on time-consuming low-level simulations. The implementation of the ANN comes to increase the functional models of AMS blocks regarding information about their transient power consumption, where this one is trained and then translated into a behavioural modelling language that is tolerant to industrial circuit simulators. All of this, without the need to have manual interaction. To validate this approach, a low power relaxation oscillator was used, and the results confirm that the energy consumption has come close to the considered time range, having an error of only 2.7%, while the needed simulation time has diminished.

2.5 Conclusion

As seen, ML and ANNs have been successfully used in the field of analog IC design automation. The Table 2.4 sums up all the contributions and applications discussed along this Section. While having the simulator as the evaluation engine is beneficial in both generality and accuracy, it suffers from demanding execution times. Researchers have been using ML to reduce the workload of the simulator, and both SVM and ANNs have shown to be very useful when optimizing simulation-based techniques.

In [15] and [18] SVMs try to reduce the number of points simulated, and therefore the workload of the simulator, however, it is hard to tune the hyperparameters on SVM and to choose the right kernel. Furthermore, the poorly performance when handling big datasets and high number of features, makes ANNs the option chosen in this work. In [30], [31], and [32], ANNs are used to replace the simulator, and even though the results show increasing evaluation speeds, there is a loss in accuracy and an excessive use of relevant resources. Thus, this makes the choice to use ANNs in this work, with the intent to enhance the optimization-based sizing, in an attempt to complement the circuit simulator used.

Table 2.4 - Summary of Related Work Techniques used to automate analog circuit sizing

Reference	Application	Method	Overview
[12], 2016	Simulation-based Evaluation	HSPICE and CALIBRE	Automation circuit design by automating circuit synthesis and layout
[20][21][22][23][24][25], 2012 - 2019	Simulation-based Evaluation	Electromagnetic, ELDO, HSPICE and Post-layout simulations	Optimization of RF circuits using a simulator as evaluation engine
[15], 2013	Simulation-based Enhanced by SVM	SVM classifier	SVM used in multiobjective design method to automate synthesis circuit
[28], 2005	Simulation-based Enhanced by SVM	SVM classifier	SVM used to identify the feasibility design space, eliminating many points of the design space
[29], 2003	Simulation-based Enhanced by ANN	ANN	ANN used to create fast and efficient models to estimate performances parameters
[30], 2020	Simulation-based Enhanced by ANN	DNN	DNN used to replace SPICE and reduce the synthesis time
[31][32], 2002 - 2003	Simulation-based Enhanced by ANN	ANN	ANN used to automate the design flow and create models for large analog design space
[33], 2019	Simulation-based Enhanced by ANN	ANN	Determine circuit performances
[34], 2019	Simulation-based Enhanced by ANN	DNN	Reduces the number of simulations, boosting the optimizer's sample efficiency
[35][36], 2018 - 2019	Predicting devices size	DL and ANN	Generates a model that maps specifications to sizings
[37], 2008	Predicting devices size	ANN	Determines the sizes of transistors by the designer specifications
[38], 2015	Predicting devices size	ANN	Determines parameters values, by using several ANNs
[39], 2017	Predicting devices size	ANN	Prediction of an amplifier sizing, by using only a small amount of data on the testing phase
[43], 2019	Other Application	TDNN	Enhances functional models of AMS blocks

Chapter 3

Architecture and Implementation

The work proposes the development of ANNs, namely DNNs, to enhance the current analog RF IC sizing optimization methodologies, given the sizing of the devices. The goal is to reduce the total effort of the evaluation engine, i.e., the simulator, by reducing the number of candidate circuit sizing solutions that are actually simulated. This is accomplished with the use of two ANNs, one classifier, and one regressor. As this work focuses on VCO circuits, the combination of these two will try to predict whether a certain solution can generate all the performances metrics, i.e., if it will “converge”, and when this is the case, at what frequency does the circuit oscillates.

The optimization process is described in Figure 3.1, where the DNNs play the filter role of selecting from the candidate circuits the most likely to be useful to be presented to the simulator.

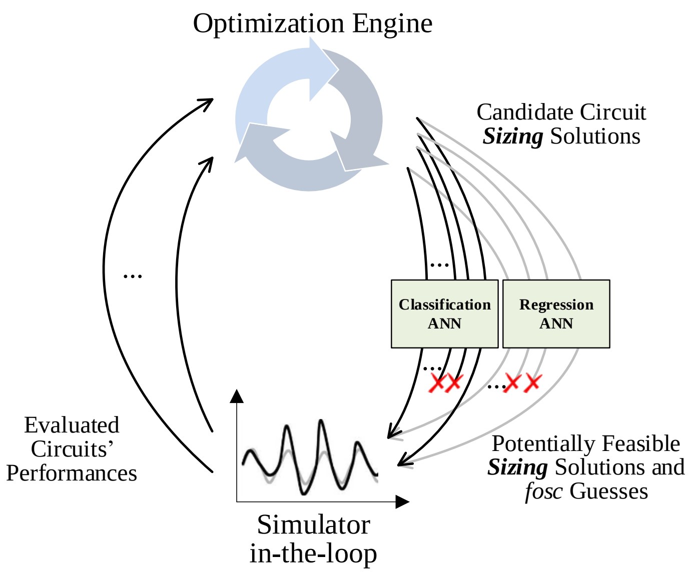

Figure 3.1 - Proposed Optimization

3.1 Circuit used and dataset definition

The development of the ANNs will be used, initially, to accelerate the optimization sizing process of a complex dual-mode class C/D VCO described in [10], where a schematic of the circuit is shown in Figure 3.2.

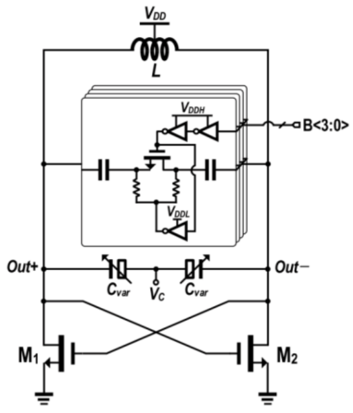

Figure 3.2 - Dual-mode class C/D VCO schematic reprinted from [10]

This circuit contains 43 devices, and there are 28 optimization/design variables used in [10], such as devices' widths, lengths, and number of fingers, whose sizes must be chosen in order to meet the desired circuit performances. The list of the optimization variables is shown in Table 3.1.

Table 3.1 - Optimization variables

Variable	Units	Min	Grid	Max
Ind_radius	µm	15	5	90
Ind_nturns	-	1	1	6
Ind_spacing	µm	2	1	4
Ind_width	µm	3	1	30
mccl, m1l	nm	60	20	240
mccw, m1w	µm	0.6	0.2	6
mccnf, m1nf	-	1	1	32
mccm	-	1	1	100
moscapw	µm	0.4	0.2	3.2
moscapl	µm	0.2	0.2	3.2
mimvw, mimvl, mim1w	µm	2	0.2	20
r1l, r2l, r3l, r4l	µm	1	0.2	10
r1m, r2m, r3m, r4m	-	1	1	20
nfn1, nfn2, nfp1, nfp2	-	1	1	100

These optimization variables (which impact the sizing of the different devices) will be used as inputs for the ANNs, being these ones evaluated for 9 different tesbenches variations, i.e., typical and extreme conditions. These different testbenches for extreme conditions are called PVT corners and are outlined in Table 3.2.

Table 3.2 – PVT Corners considered

Name	Process	Voltage	Temperature
tt	TT	0.35V	25°C
ff	FF	0.35V	25°C
fs	FS	0.35V	25°C
sf	SF	0.35V	25°C
ss	SS	0.35V	25°C
300mV	TT	0.3V	25°C
400mV	TT	0.4V	25°C
M40dC	TT	0.35V	-40°C
85dC	TT	0.35V	85°C

Each sizing combination will be simulated for each PVT corner and, a worst-case tunning range optimization is considered. The circuit is desired to oscillate between 3.9GHz and 4.8GHz, therefore the circuit will be optimized for two tunning modes, b0000 and b1111, for the values of 3.9GHz and 4.8GHz, respectively, generating for each simulation the performances illustrated in Table 3.3. This will result in 18 simulations, 9 for each tunning mode, where each one produces 10 performances values.

Table 3.3 - Performances considered

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

3.2 Feature Engineering

When developing a ML system, one of the main challenges is the process of extracting features from the raw input data. This is where feature engineering arises, having as its main goals the preparation of the proper input dataset, according to the ML algorithms requirements, and the improvement of the models created.

When training an ANN, the system can only have satisfactory results if the training data contains enough relevant features. Feature selection is the process of selecting only the most important features from the dataset, taking out the non-relevant ones.

The scale and distribution of the data are also very important to bear in mind since this may be different for each variable. Some features may have different scales while others may not even have units. These variations may be an issue during the training stage when performing updates to the parameters. To solve this, it can be applied to some scaling methods such as min-max and standardization.

Min-max scaling makes a certain unscaled feature have a value ranging from 0 to 1. This is obtained through 3.1, where x is the feature and *x'* the resulting scaled feature:

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

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

$$x' = \frac{x - \mu}{\sigma} \tag{3.2}$$

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

On top of that, it is also necessary to prepare the dataset for each ANN. The tunning mode must be added to the input features, and for the regression ANN the PVT corner as well. Also, for the classification ANN, there is the need to label each PVT corner, corresponding to a binary classification of “converges” or not. To do this, the 10 performances for each PVT corner will have to be grouped up and according to the values of the performances the group will be labelled as “converge” or not, i.e., if there is at least one value that is equal to 0 or non-defined then it is labelled as “not converges”, otherwise it is labelled as “converges”. This process of data preparation must be done before training the ANNs.

3.3 ANNs structure and training

For this work, the two ANNs that are going to be developed have similar architectures, with some small differences.

The classification will have, *n* neurons, the device sizings and the tuning mode, and *y* neurons as output for each Process Voltage Temperature (PVT) corner considered, as shown in Figure 3.3. With these inputs, the ANN will be able to predict if a given candidate circuit sizing solution should be simulated.

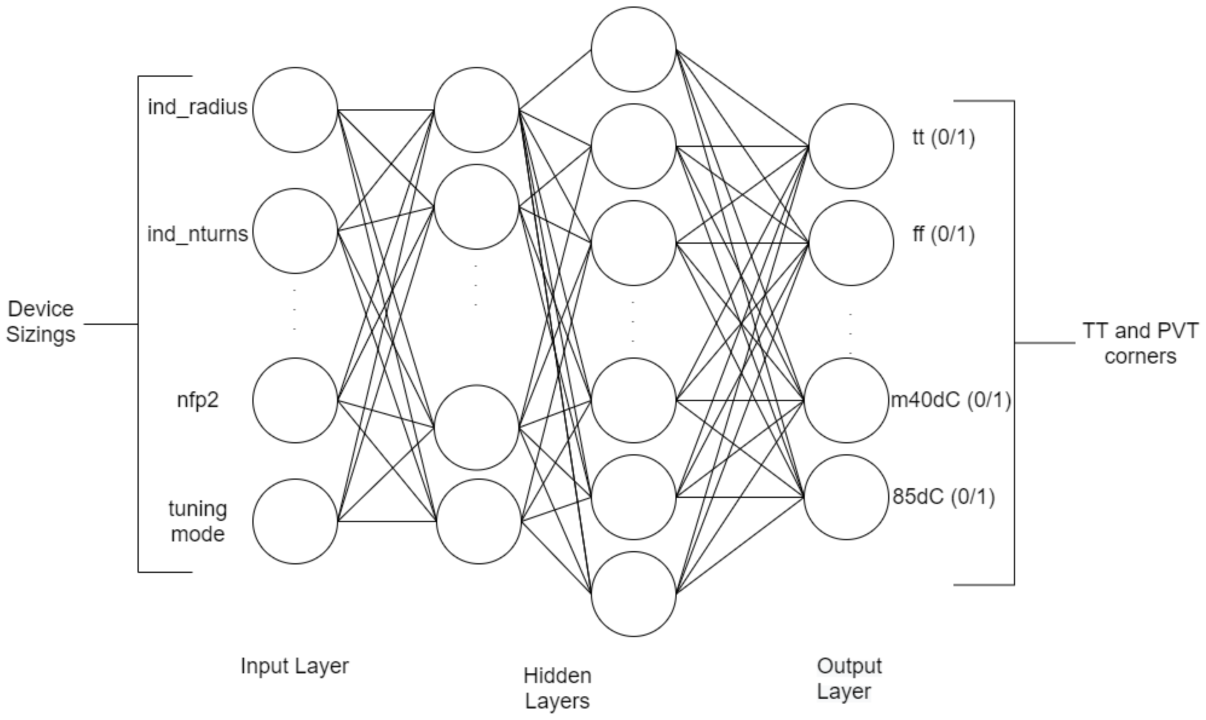

Figure 3.3 – Structure of the Classification ANN

Whereas the regression, besides having the same inputs as the classification, have one additional input, the corner to be considered. The output will determine the oscillatory frequency for that specific corner and tuning mode, as shown in Figure 3.4.

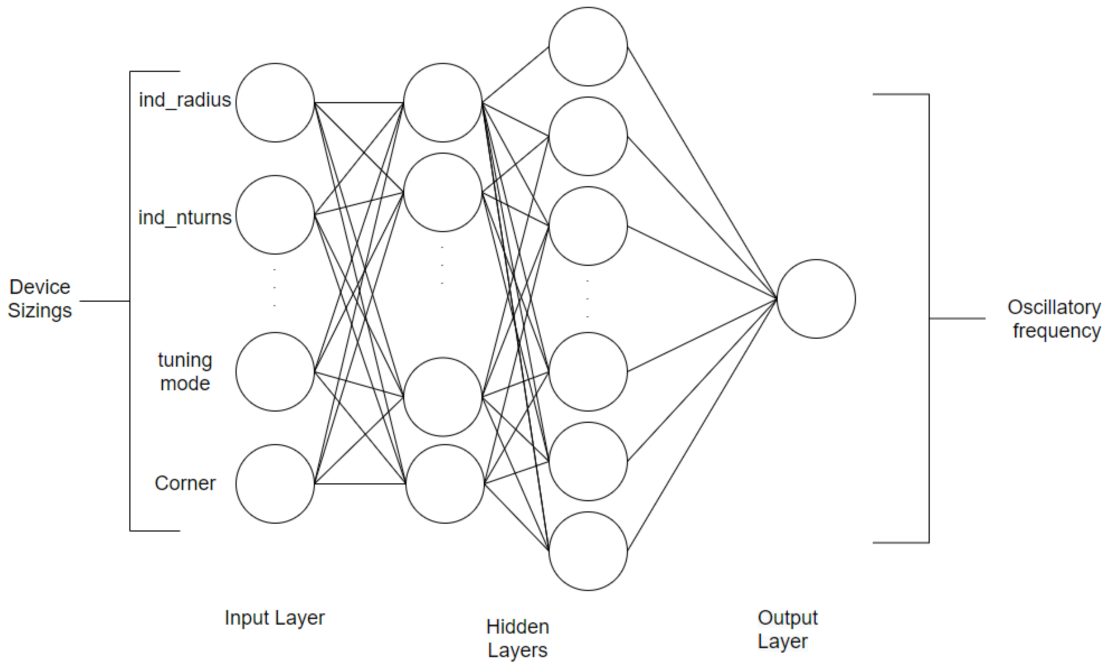

Figure 3.4 – Structure of the Regression ANN

The output may be represented by the general function, where the selection of the activation function and the update of the adequate weight values represent the learning process:

$$Y = \phi\left(\sum_{i=1}^{n}(w_i * x_i) + b\right) \tag{3.3}$$

ϕ is an activation function, which is a hyperparameter that will be further detailed and discussed in Section 3.3.3, w the weight of each neuron link, *n* the number of inputs, and b, the neuron's bias, which is often associated with an extra bias feature, normally corresponding to $\mathrm{x}_0 = 1$, with an associated weight of w_0.

Each ANN will produce a different output. The classification receives the device sizings and the tuning mode and will do a classification of "converges" or "not converges", which is the classification of each possible candidate circuit sizing solution. Each output is classified based on the values of the performances produced, for each corner considered. If the values produced are all different than 0 it is classified as "converge", otherwise as "not converge" and discarded.

The regression receives the same inputs, plus the corner to be evaluated, and the output produced is a prediction of the oscillatory frequency. This regression will be executed only for the solutions that are classified as "converge", since there is no point on predicting the oscillatory frequency for solutions that will not be simulated.

The target solution will have a general behaviour for the circuit analysed, independently of the designer constraints, since the solution is only dependent on the device sizing information. The trained solution will be able to be reused on all different optimizations for the same circuit.

3.3.1 Training the ANNs

When training the ANNs, in every instance, the algorithm measures the network's output error, the difference between the target value and the predicted ones and computes the contribution that each neuron in the last hidden layer had to each output neuron's error. Then, the impact of the error from each neuron in the previously hidden layer is measured, until the algorithm reaches the first layer, the input layer. This logic of following the reverse path is done to compute the error gradient across all the connection weights, by propagating the error gradient backward. The error is evaluated via a defined loss function, *L*, that in order to be minimized the weights must be updated, by defining the error's derivative $\frac{dL}{dwij}$, where i is the weight associated to the neuron *j*.

When choosing the loss function, one must choose to which problem it refers to. Since this work will classify points into two classes, a variation of categorical crossentropy will be used, where the loss function is equal to the average of the categorical crossentropy, called binary crossentropy, calculating the loss by using 3.4.

$$L = -\frac{1}{n}\sum_{i=1}^{n} y_i * log\,\hat{y}_i + (1 - y_i) * \log(1 - \hat{y}_i) \tag{3.4}$$

On the other hand, when considering regression, where the value of the frequency is predicted, Mean Squared Error (MSE) is the most appropriate function to use. After arriving to a frequency value, the average squared difference between the real frequency and the one predicted is calculated, being the loss calculated by 3.5.

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

The size of the dataset influences the approach and number of examples to be used in the training of the model and the calculation of the loss in each iteration. Having datasets that can go from 26000 to 92000 examples, the approach that will be addressed is to use mini-batch gradient descent, where the number of batches is chosen, performing the model's gradient on small random sets of instances. Having no exact formula that says the perfect batch size, the approach that will be conducted is to start by using small batches and gradually increase its number.

In order to minimize the cost function and finding optimal values for any model's parameters, the optimization algorithm with more potential is Adam. Adam has three hyperparameters which are showcased in 3.6, 3.7, and 3.8, respectively.

$$m_{t+1} = \frac{\beta_1 m_t + (1-\beta_1)\nabla L(w_t)}{1-\beta_1} \tag{3.6}$$

$$v_{t+1} = \frac{\beta_2 v_t + (1-\beta_2)\nabla L(w_t)^2}{1-\beta_2} \tag{3.7}$$

$$w_{t+1} = w_t - \frac{\eta m_t}{\sqrt{v_t + \epsilon}} \tag{3.8}$$

Even though it has three hyperparameters, these ones have default values that tend to perform well [44], therefore, there is no need to tune them. The default values typically correspond to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and ϵ to a small value such as 10^{-8}.

Adam works well for disperse data, has a fast convergence rate, and it usually finds the optimal solution. However, if needed as an alternative optimization algorithm, RMSProp will be considered. The training is briefly illustrated in Figure 3.5.

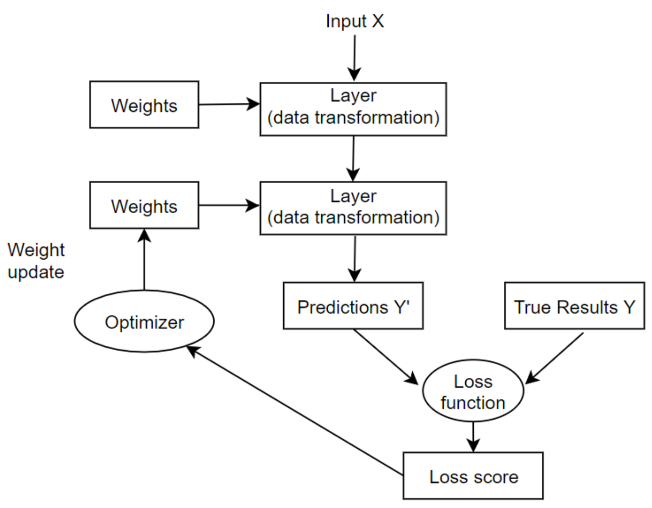

Figure 3.5 - ANNs Training Scheme

3.3.2 Validation

To assess the performance of the model it is necessary to do a model validation. This assessment will be executed by splitting the dataset into train, validation, and test data. On the training phase, the fitted model is used to predict the responses for the observations in a portion of the training set called the validation dataset, providing an unbiased evaluation of the model. Finally, the test set is used to evaluate the final model fit.

One problem that may occur when training, is overfitting and underfitting. Overfitting occurs when a model is over adjusted to the given training data. On the other hand, underfit occurs when the model is not complex enough. For a model to be good, it must have a good generic relationship between the input and output data. Thus, to tackle the issue of overfitting, if identified, some regularization techniques will be used, such as early stopping and dropout.

Early stopping is an efficient option to avoid overfitting, by interrupting the training when the performance starts to decrease on the validation sets. As iterations are made, the model generated learns and the prediction error on the validation set gets lower. This method is shown in Figure 3.6, where after a certain amount of training steps the model starts to become overfit.

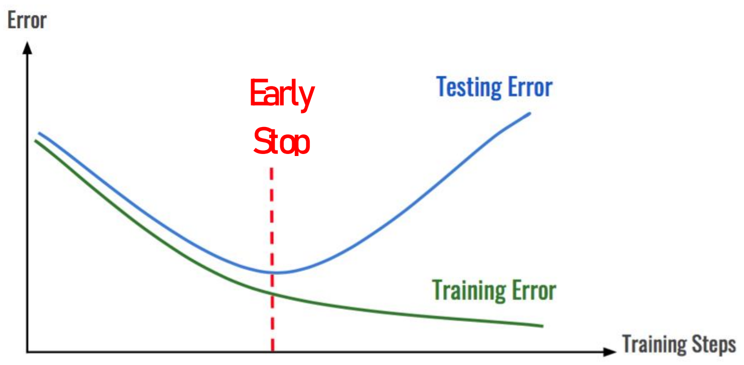

Figure 3.6 – Early stopping reprinted from [45]

Dropout has proven to perform efficiently [46], showing results of 2% accuracy boost, which in a model that already has 95% accuracy corresponds to an almost 40% reduction in the error rate. For each training step, every neuron, except for the output ones, has a probability *p* of being momentary dropped from the network. This means that for one specific step, the dropped neurons are enabled to receive and produce connections, being available again for the next steps. Dropout tends to take some time to find convergence, but the resulting model, when tuned accordingly, turns to be well worth the time and effort.

3.3.3 Model tuning

Even though ANNs are very flexible, one of their major disadvantages is the number of hyperparameters to be tuned. In this section, the most important ones and the ones that will be tuned will be discussed and the approach to finding the best values for each one will be addressed.

Hidden layers

The complexity of the ANN is influenced by the number of hidden layers used. As more hidden layers are used, more does the complexity of the model increases, which often results in good results, but also does the model tend to overfit the training set. Additionally, it also increases the time and computational effort for the training stage. As there is no formula to choose the right hidden layers, to begin, one single hidden layer will be implemented, and they will be incremented until the accuracy keeps going up, until the point where it cannot get higher, meaning that the model generated overfits.

Number of Neurons

Obviously, the number of neurons in the input and output layers is determined by the number of features and the type of output pretended. In this work, the number of inputs will be the number of the devices sizings, plus the tunning mode, and one output for each corner analysed, for the classification. The regression will have one more input, a corner. The output will be the prediction of the oscillatory frequency. And thus, for the VCO shown in Figure 3.2, the number of inputs for the classification and regression, will be 29 and 30, respectively, with 9 and 1 outputs, respectively, corresponding to a classification and a regression for each TT or PVT corner, as shown in Table 3.2.

Having this established, the hyperparameter to be tuned is the number of neurons in the hidden layers. The first hidden layer will have *Z* neurons and then follow a common practise that consists of decreasing the number of neurons for each layer that follows, where many low-level features are merged into far fewer high-level features.

However, another approach that can be used is to use the same number of neurons for all hidden layers, reducing this way the complexity of tuning this hyperparameter. Since there is not a formula that says the perfect number of neurons per layer, the approach that will be conducted is to gradually increase the number of neurons until the point where the model starts overfitting.

Learning Rate

One important hyperparameter is the learning rate. Learning rate dictates how fast does the model respond to the estimated error in each weight update and the ability to converge to a minimum. Setting this value is a hard task, having a high value makes the model rapidly adapt to new data but rapidly forgetting about the old one, on the other hand, having a low value makes the system learn more slowly, nonetheless makes this one less sensitive to noise in the data.

The process of choosing the most appropriate learning rate is not a very well defined one, but rather a trial-and-error, where the approach to follow is, to begin with, a high value and decrease this one until reaching a point of no improvement.

Activation Functions

An activation function decides if a neuron should be activated or not. In regression, when obtaining the frequencies, where the output is a real value, it will be used as the activation function a linear function $f(z) = z$, illustrated in Figure 3.7.

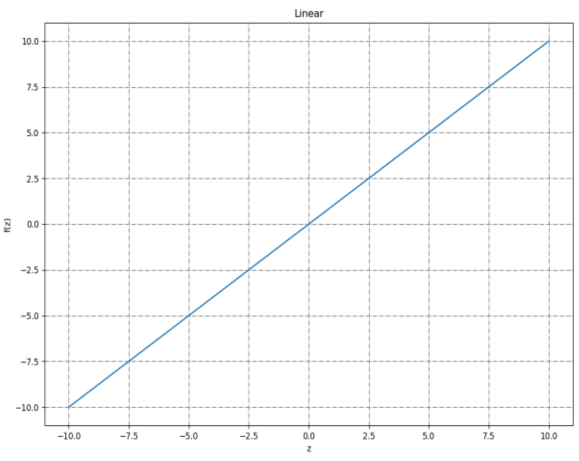

Figure 3.7 - Linear function

However, for the output of the classification and the hidden layers of both the ANNs, it is necessary to have activation functions that produce non-linearity in the output of a neuron, since the neuron does not know the bounds of its output which can reach values ranging from -∞ to +∞.

Regarding the classification, the sigmoid function, shown in Figure 3.8, is a prominent function to be used, having the ability to map the input values of a neuron to a value that goes from 0 to 1. This function solves cases of values that reach values too high, by setting a limit, being a good function to use in classification problems.

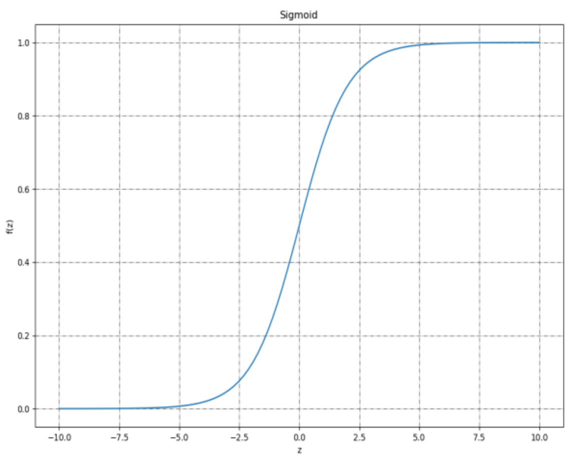

Figure 3.8 - Sigmoid function

As discussed in [47], there are other good candidate activation functions that surpass the sigmoid function when addressing DNNs. One is a variation of the Rectified Linear Unit (ReLU) called leaky ReLU, illustrated in Figure 3.9. This one solves the dying ReLU problem, that happens in some cases where neurons stop outputting values other than 0, since when the gradient of the ReLU function is 0, it will no longer respond to changes in the error. It is computed through $f(z) = max(0, z)$, where no saturation occurs for positive values. Having this said, this will be the function to be used on the hidden layers for the regression.

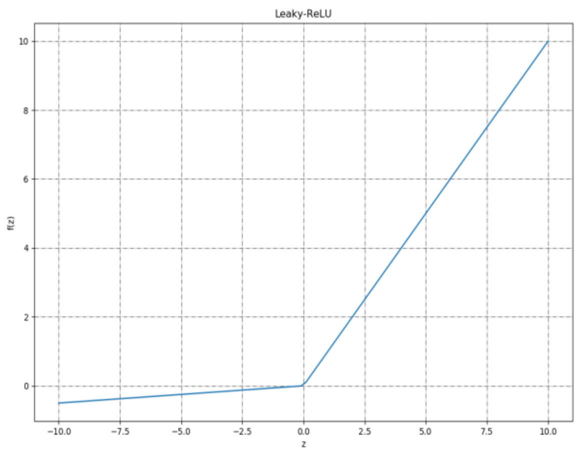

Figure 3.9 - Leaky ReLU function

3.3 Conclusions

In this chapter, a carefully study of the implemented models was exposed. The circuit to be optimized, a Class-C/D VCO, was presented, and the dataset defined, where each optimization/design variable is showed and the conditions for each simulation introduced. Furthermore, the classification and regression ANNs were presented, it's hyperparameters presented and explained in detailed, as well as the choice of the loss functions and activation functions to test and use. In the next Section, the results of the training of the ANNs will be analysed, and the training further explored, step by step.

Chapter 4

ANNs Training

In this chapter, different ANN approaches are analysed and tested. Two circuits were used as a case study, where for each one was built a classification and a regression ANN, and its different hyperparameters tuned. All ANNs were implemented using Python language with the use of different ML libraries such as TensorFlow [48] and Keras [49]. The code was executed on an Intel® Core™ i5-8600K 6 cores CPU 3.60 GHz with 16 GB of RAM.

4.1 Class-C/D VCO

This section describes how the dataset is studied and dealt with, and how the models arise from this study and how its results are maximized. Firstly, the results of the models for the class-C/D VCO circuit are presented. This circuit is optimized for Ultralow-Power IoT and Ultralow-Phase-Noise Cellular Applications, as previously presented on Section 3.1.

4.1.1 Dataset analysis

Analysing the dataset provided for this circuit, some pre-processing had to be done. The dataset contains 92115 rows of data, where for each one, it is given a combination of the 28 device sizings and its resulting 180 performance values, 90 for each tuning mode. To start off, following the logic from the previous chapter, for both classification and regression, the tuning mode is considered as an input, therefore, each possible combination of device sizings is replicated and a column is added containing the tuning mode considered, resulting on twice as many rows, i.e., 184230 rows of data. Then, the following actions differ for each ANN:

- **Classification:** For the classification, in each row, the 10 values of performances generated for each corner are analysed, in order to be label the corresponding corner as "converge" or "not converge". To do so, each corner is examined and if it has a value different than *NaN* for every performance metric, then this one is labelled as "converge", otherwise "not converge". Having done this for every row, the dataset for the ANN is ready, arising on a total of 184230 rows and 38 columns, 28 device sizings and the tuning mode for the inputs, and the 9 corners for the outputs. This process is illustrated in Figure 4.1;

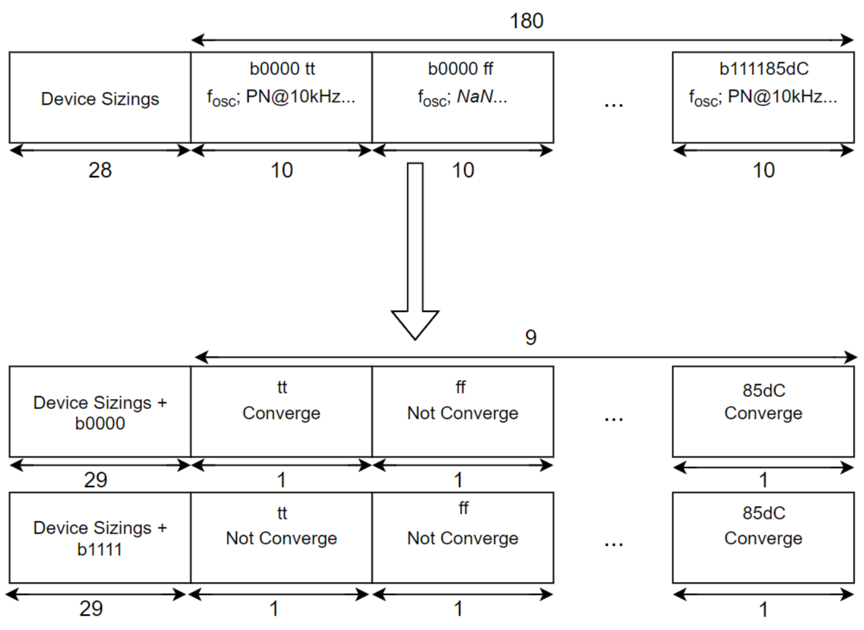

Figure 4.1 – Pre-Process of Classification Dataset

- **Regression**: Additionally, for this ANN the dataset had to go through more actions. Since this ANN will only be used in a specific event in case the Classification assumes that a certain corner converges, this one only needs to be fed information about data that is labelled as converges. Bearing this in mind, each corner is considered as input, therefore, each row of data is replicated, but in this case 9 times, so that for each possible combination of sizing variables there is an oscillatory frequency guess, as it is shown in Figure 4.2. Finally, after a careful analysis, the data had to go through a final procedure, the remove of outliers. After this step, the dataset is ready to be used for training the ANNs. Table 4.1 showcases the data and its details having outliers, and after having these ones removed.

Table 4.1 – Frequency distribution with and without outliers

	Outliers	Without outliers
Max. oscillatory frequency	1.63E+16	6.47E+09
Min oscillatory frequency	-1.03E+02	1.07E+09
Standard Deviation	1.71E+13	7.93E+08
Mean	4.16E+10	4.21E+09
Quantile 0.25	3.58E+09	3.59E+09
Quantile 0.5	4.14E+09	4.14E+09
Quantile 0.75	4.89E+09	4.88E+09

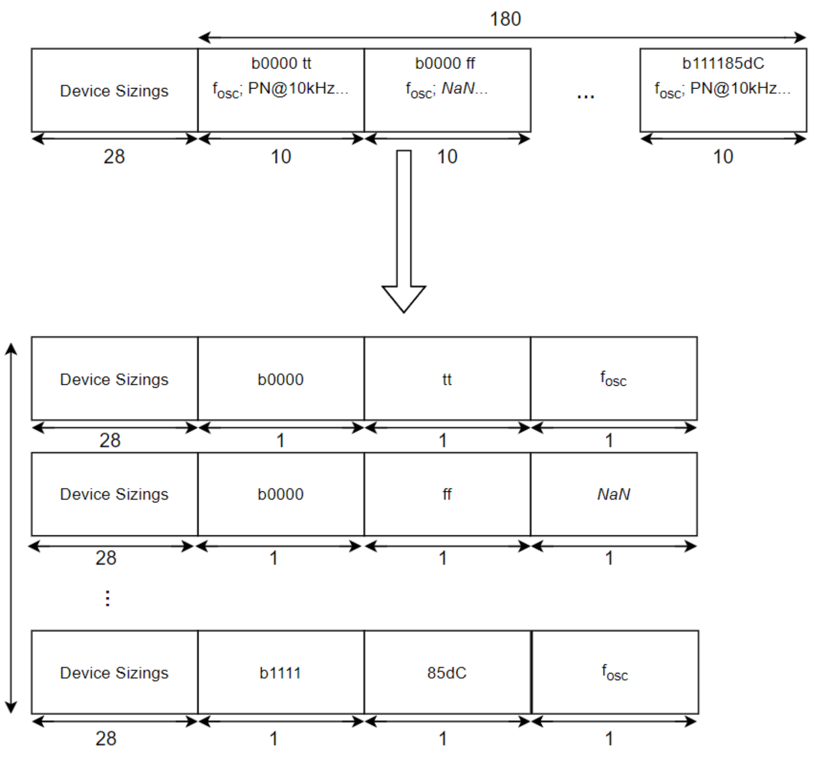

Figure 4.2 - Pre-Process of Regression Dataset

4.1.2 Classification ANN study

Having the dataset prepared, it can be used to program and train the ANNs. Firstly, the data is divided into train and test data with a ratio of 80% and 20%, respectively, and in order to be sure that the results are consistent with different data, this is firstly shuffled, randomly. The metrics used to evaluate the model were:

- **Loss**, which results on the difference between the real value and the predicted one;
- **Binary accuracy** which computes how often predictions equal labels. For example, a value of 50% means that for the 18 PVT corners, the model is able to correctly predict the state of half of them, i.e., if they converge or not;
- **Recall** which states what proportion of actual positives were identified correctly. In this case, the value represents the amount of correctly identified corners that converge, or not converge.

Tuning the different hyperparameters, as part of the training process of the model, is a critical process to achieve the best model. Even though there is interaction between hyperparameters when tunning an ANN there is no established order to follow and each of the hyperparameter can be changed independently. The order followed in this work is illustrated in Figure 4.3.

As a starting point, learning rate is the first hyperparameter to tune since is one of the most important factors as it will deeply impact the number of experimentations [50]. The batch size is the one succeeding, since that will largely affect the time of training on future experiments. Following, the activation functions and the number of hidden layers, and number of neurons per layer should be tunned as well. To ensure the best model structure possible, and to cease, the normalization is the last one to tune. This one does not have a strong dependency with the remaining ones, and therefore, it can be tunned as the last one.

The initial hyperparameters were tuned with a batch size as 32, ReLU as the activation functions for the hidden layers and sigmoid for the output ones, a simple architecture of an ANN with two hidden layers with 200 neurons per layer and MinMax with a range going from 0 to 1 as the normalization technique.

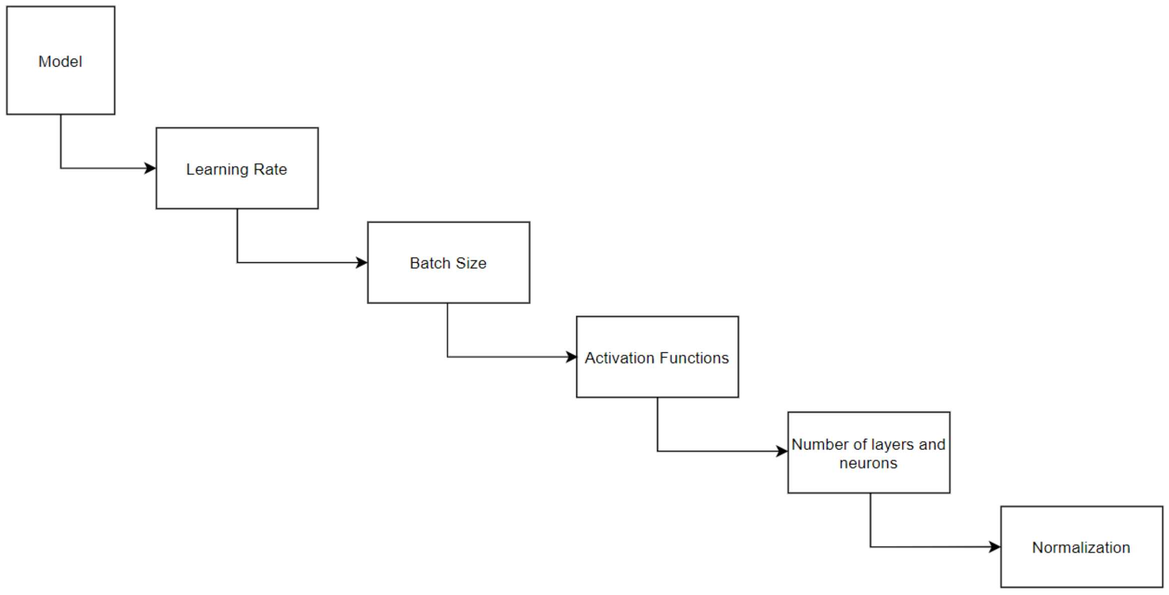

Figure 4.3 – Order of the tuning hyperparameters

Learning Rate

The learning rate controls how much does the model changes in response to the estimated error, therefore the higher it is the faster the model changes and the faster it moves toward a minimum of a loss function. However, if the value is too high the minimum of the loss function can be lost, hence one needs to start with a relatively small value and then start decreasing it.

The results on Table 4.2 shows that the value of 0.003 is the one with the lower training and validation loss and highest training and validation accuracy. Even though the values of the training and validation recall are not the highest, achieving values of 99.13% and 98.87, respectively, the overall results make 0.003 the optimum value.

Table 4.2 – Learning Rate evolution with 32 batch size, ReLU and sigmoid activation functions, two hidden layers and 200 neurons per layer, and MinMax of 0 to 1

	Training Loss	Validation Loss	Training Binary accuracy	Validation Binary accuracy	Training Recall	Validation 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%

Batch Size

As mentioned in chapter 3, the approach used to tune this hyperparameter would be to use mini-batch gradient descent, where the number of batches used would be neither the total number of samples, neither just one, but instead a value that would be in between these two.

As the Table 4.3 showcases, with the use of a 256 batch size the model achieves the lowest validation loss, and also the highest validation accuracy. Even though, it doesn't surpass other values in the other metrics, for example, the training loss of using a 128 batch size is only 0.1017, the values of 0.1055 and 98.79% in validation loss and accuracy, respectively, and also the fact that the use of a bigger batch size reduces the time of training, makes the value of 256 batch size being the one chose to be used.

Table 4.3 – Batch Size evolution with 0.003 learning rate, ReLU and sigmoid activation functions, two hidden layers and 200 neurons per layer and MinMax of 0 to 1

	Training Loss	Validation Loss	Training Binary accuracy	Validation Binary accuracy	Training Recall	Validation Recall
32	0.1148	0.1176	0.9615	0.9604	99.07%	91.09%
64	0.1219	0.1276	0.9559	0.9546	99.51%	99.47%
128	0.1017	0.1064	0.9660	0.9641	98.49%	98.38%
256	0.1025	0.1055	0.9653	0.9642	98.79%	98.73%
512	0.1046	0.1090	0.9641	0.9627	84.28%	98.10%

Activation Functions

When choosing the right activation functions, one must consider both the hidden layers and the output layers. Table 4.4 shows the results regarding the use of different activation functions in the hidden layers, where Table 4.5 shows the results in the output layers.

The results, show that for the hidden layers the sigmoid function, achieves slightly lower values in both the training and validation recall, with values of 99.10% and 98.90%, as all other functions surpass this one, apart from the ReLU. However, the best results, both in loss and in accuracy, are achieved by this function, making it the best function to use in the hidden layers. As for the output layers, the same is observed as the sigmoid function only fails to perform better in the recall metric, with a margin that does not reaches 1%, both in training and validation, which makes the sigmoid the function to use also for the output layers.

Table 4.4 – 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 to 1

	Training Loss	Validation Loss	Training Binary accuracy	Validation Binary accuracy	Training Recall	Validation 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 4.5 - Output Layers Activation Functions evolution with 0.003 learning rate, 256 batch size, two hidden layers and 200 neurons per layer and MinMax of 0 to 1

	Training Loss	Validation Loss	Training Binary accuracy	Validation Binary accuracy	Training Recall	Validation Recall
Sigmoid	0.0859	0.0920	0.9722	0.9701	99.10%	98.95%
SoftMax	1.9660	1.9679	0.0958	0.0958	99.77%	99.78%

Number of layers and neurons

The number of neurons to tune, obviously only apply to the ones present in the hidden layers, since the ones in the input correspond to the number of sizings plus the tunning mode, and the output to the number of corners. The approach was to increase the number of layers and neurons per layer gradually until the best results are obtained. As the Table 4.6 shows, when using three layers with 200, 200 and 250 neurons, respectively, the best values of loss and accuracy are achieved, only failing behind in recall, with values of 99.11% and 98.95%, for training and validation, respectively. The model will use three layers with 200, 200 and 250 neurons, respectively.

Table 4.6 – Number of layers and neurons evolution with 0.003 learning rate, 256 batch size, sigmoid activation functions, and MinMax of 0 to 1

	Training Loss	Validation Loss	Training Binary accuracy	Validation Binary accuracy	Training Recall	Validation 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%

Normalization

Finally, different ranges of MinMax and its use were tuned and compared, arriving at the results shown in Table 4.7. 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 margins of the values of recall when using different ranges, not too problematic, and therefore the choice of using MinMax with the range mentioned.

Table 4.7 – 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	Validation Loss	Training Binary accuracy	Validation Binary accuracy	Training Recall	Validation 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%

4.1.3 Classification ANN final model

The training process led to a model with its tuned hyperparameters described in Table 4.8 and the loss and accuracy of both training validation shown in Figure 4.4 and Figure 4.5. By analysing the loss graphs as the epochs start increasing, the loss curve starts decreasing achieving its best result on the last epochs. At this point, the training curve starts to get lower than the validation one, hence the need to stop rests there, otherwise the model would start to overfit. As for the accuracy graphs, the same results are observed, as the epochs increase the value gets better, in this case it goes higher, which means that the model learns to predict the status of the corners more correctly. This is proven by the value of the binary accuracy, meaning that the model only fails to predict a very low number of cases, less than 5%, where a corner does converge or not. Therefore, the probability of the model to discard a valuable point or use a non-one, is low.

Table 4.8 – 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)

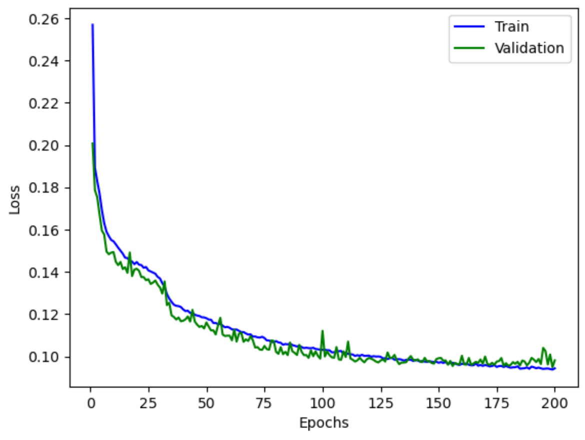

Figure 4.4 – Training and validation loss

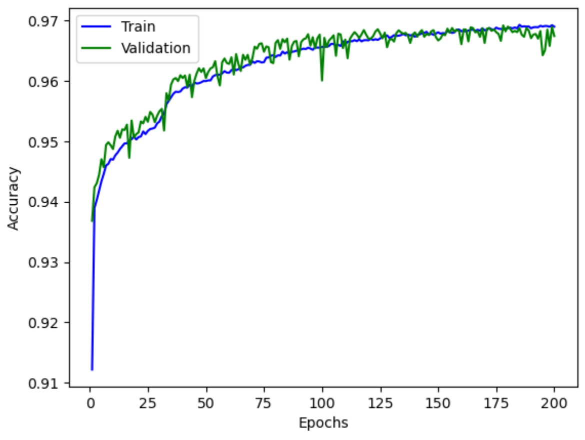

Figure 4.5 - Training and validation accuracy

Furthermore, to emphasize the validity of the model two more metrics were used, precision and F1 score. Precision shows the portion of correct positive identifications, while F1 is an overall measure of a model's accuracy that in case of a higher value, it means that there is a low number of false positives and false negatives, hence showing that the model is correctly identifying the classes. The formula of precision and F1 score are shown in 4.1 and 4.2, respectively, and its results on Table 4.9.

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Posivites} \tag{4.1}$$

$$\text{F1 score} = \frac{2 * True\ Positives}{2 * True\ Positives + False\ Posivites + False\ Negatives} \tag{4.2}$$

Table 4.9 – Train and Test Precision and F1 score results

	Train	**Test**
Precision	0.979	0.977
F1 score	0.985	0.983

4.1.4 Regression results

The training process is almost the same as the one considered for the classification, however this ANN needs to be more accurate since it is trying to predict a real value, the oscillating frequency. Therefore, more metrics were used and a deeper study to the results was made. The metrics used to evaluate the model were:

- **Loss**, that results on the difference between the real value and the predicted one by using the function MSE;
- **Mean Absolute Error (MAE)**, which gives the mean of the absolute values of the prediction errors;
- **Mean Absolute Percentage Error (MAPE)**, a measure of how accurate the model is.

As a starting point, the same logic was used as before, therefore the hyperparameters were tuned with a batch size 32, leaky ReLU as the activation functions, MinMax with a range going from 0 to 1 as the normalization technique and two hidden layers with 200 neurons per layer.

Learning Rate

Following the same logic, the learning rate started as a relatively small value and then gradually decrease it until finding the best value. Analysing the Table 4.10, the best values of MSE and MAE come from the use of a value of 0.0008, and so this value was the one chosen.

Table 4.10 – Learning Rate evolution with 32 batch size, leaky ReLU activation functions, two hidden layers and 200 neurons per layer, and MinMax of 0 to 1

	Training MSE	Validation MSE	Training MAE	Validation MAE	Training MAPE	Validation 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

Analysing the results, it is possible to observe that the values of the MAPE were too high, and therefore the order of the tuning of the hyperparameters could not follow the order used before, hence normalization had to be taken in consideration immediately.

Normalization

After inspecting the results obtained from using and changing the parameters of MinMax, these ones rapidly changed. As Table 4.11 shows, the best results of MSE and MAE come from using MinMax with a range of 0 to 1. However, the MAPE values are too high, and so, MinMax was used with a range of 1 to 2, since overall, it has the best results, achieving the lowest values of MAPE and MSE.

Table 4.11 – Normalization evolution with 0.0008 learning rate, 32 batch size, leaky ReLU activation functions, two hidden layers and 200 neurons per layer

	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

Batch Size

The approach used to tune this hyperparameter, was also to use mini-batch gradient descent, where one can see that the best results of all metrics, MSE, MAE, and MAPE, are obtained from using a batch size of 256, as seen in Table 4.12

Table 4.12 – Batch size evolution with 0.0008 learning rate, leaky ReLU activation functions, two hidden layers and 200 neurons per layer and MinMax of 1 to 2

	Training MSE	Validation MSE	Training MAE	Validation MAE	Training MAPE	Validation MAPE
Leaky ReLU	0.0006	0.0006	0.0131	0.0132	0.8554	0.8635
ReLU	0.0008	0.0009	0.0189	0.0190	1.2472	1.2529
Sigmoid	0.0008	0.0008	0.0175	0.0175	1.1448	1.1430

Activation Functions

As opposed to the classification ANN, in the regression it was only necessary to tune the activation function in the hidden layers, since the output layer would have a linear function, because the output values are numerical and non-negative, and do not need to suffer any transformation. Hence, different functions for the hidden layers were tested, where the Table 4.13 shows that, when using the Leaky ReLU function, the best values for every metric are obtained.

Table 4.13 – Activation Functions evolution with 0.0008 learning rate, 256 batch size, two hidden layers and 200 neurons per layer and MinMax of 1 to 2

	Training MSE	Validation MSE	Training MAE	Validation MAE	Training MAPE	Validation MAPE
32	0.0012	0.0012	0.0224	0.0225	1.4365	1.4402
64	0.0010	0.0010	0.0190	0.0191	1.2736	1.2816
128	0.0011	0.0011	0.0217	0.0218	1.4686	1.4709
256	0.0009	0.0009	0.0189	0.0189	1.2590	1.2591
512	0.0011	0.0011	0.0231	0.0231	1.5500	1.5499

Number of layers and neurons

Finally, the number of layers and neurons per layer were tuned, having arrived at the conclusion 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 showed in Table 4.14.

Table 4.14 – Number of layers and neurons evolution with 0.0008 learning rate, 256 batch size, leaky ReLU activation functions, and MinMax of 1 to 2

	Training MSE	Validation MSE	Training MAE	Validation MAE	Training MAPE	Validation 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

4.1.5 Regression ANN final model

Having tested and tuned all the hyperparameters, the ANN responsible for the regression was trained and its architecture and parameters are showed in the Table 4.15. The results of the metrics used for both the training and validation are shown in the Figure 4.6, Figure 4.7, and Figure 4.8. Analysing the graphs, the model learns to predict the oscillatory frequency, having a lower error value as the epochs keep increasing, up to the point where the model stops training since an overfit would occur if it would continue.

Table 4.15 – 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)

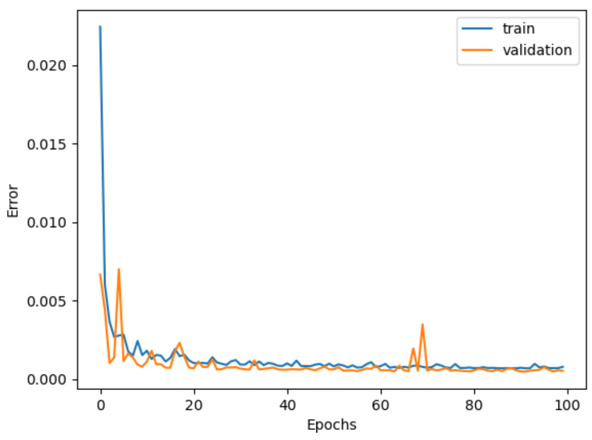

Figure 4.6 - Training and validation MSE

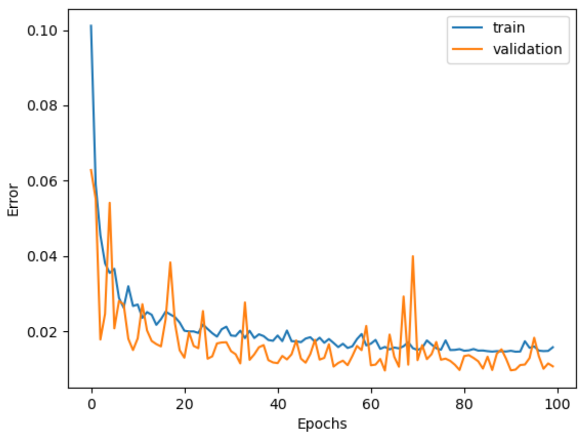

Figure 4.7 - Training and validation MAE

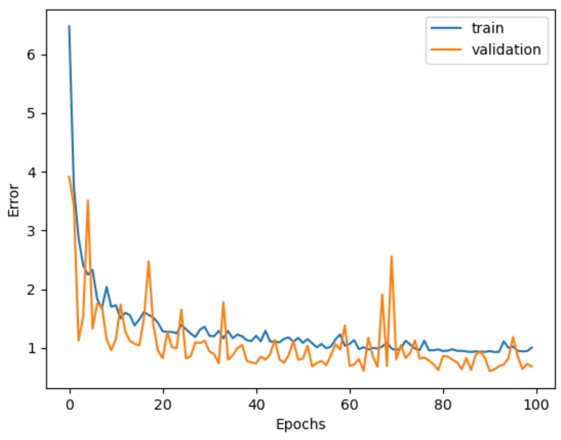

Figure 4.8 - Training and validation MAPE

Additionally, by analysing the histograms in Figure 4.9 and Figure 4.10, it is possible to see that the values predicted by the model when facing the test data, do not differ to a great extent from the real values. 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 4.16, showing that, even though the worst results are still high, the overall results show that the ANN can predict with a strong assurance the oscillatory frequency value. Therefore, it is reasonable to assume that the model's training was successful.

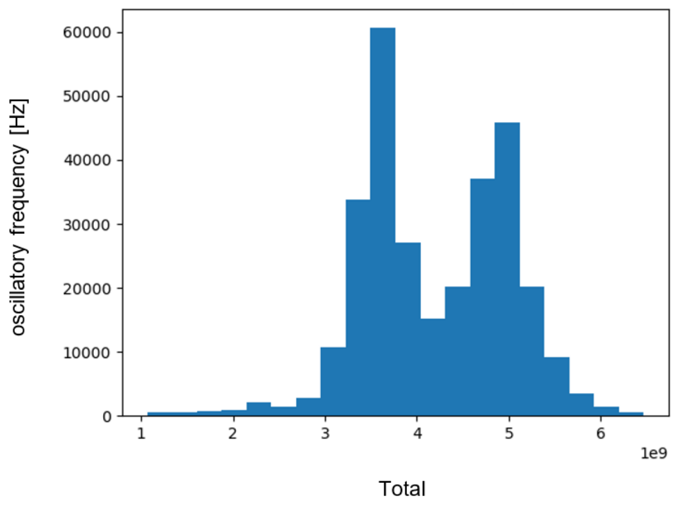

Figure 4.9 – Real Frequency Values

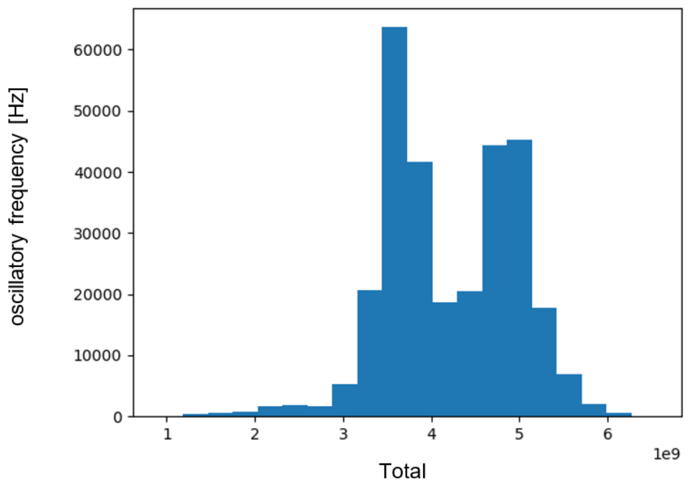

Figure 4.10 – Predicted Frequency Values

Table 4.16 – Test MAPE results and difference between predicted frequency values and real ones

	Test MAPE	Test Prediction Normalized	Test Prediction De-Normalized	Real Value Normalized	Real Value De-Normalized
Worst Values	9.22e+01	1.93e+00	6.08e+09	1.00E+00	1.09E+09
	8.68e+01	1.91e+00	5.96e+09	1.02E+00	1.17E+09
	8.60e+01	1.93e+00	6.09e+09	1.04E+00	1.27E+09
	8.41e+01	1.87e+00	5.77e+09	1.02E+00	1.16E+09
	8.22e+01	1.84e+00	5.62e+09	1.01E+00	1.13E+09
Medium Values	5.97e-01	1.47e+00	3.62e+09	1.46E+00	3.57E+09
	5.97e-01	1.73e+00	5.00e+09	1.72E+00	4.95E+09
	5.97e-01	1.77e+00	5.21e+09	1.78E+00	5.27E+09
	5.97e-01	1.67e+00	4.69+09	1.66E+00	4.64E+09
	5.97e-01	1.49e+00	3.73e+09	1.48E+00	3.68E+09
Best Values	2.08e-05	1.57e+00	4.17e+09	1.57E+00	4.17E+09
	2.00e-05	1.70e+00	4.82e+09	1.70E+00	4.82E+09
	1.69e-05	1.46e+00	3.54e+09	1.46E+00	3.54E+09
	1.65e-05	1.60e+00	4.32e+09	1.60E+00	4.32E+09
	1.64e-05	1.93e+00	5.01e+09	1.00E+00	1.09E+09

4.2 Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO

In this section, the results regarding another circuit, Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO, illustrated in Figure 4.11, are presented, following the logic that was used in the previous section. This circuit was designed for 4.2 to 5.1 GHz Ultralow-Power in a 65-nm CMOS node, having a total of 22 optimization variables, being simulated for 9 different testbenches variations [51]. For this new circuit, the objective is to use the same model as the one used before. Nonetheless, new models have also been trained with its specific hyperparameters for this circuit topology, and the results are compared with the first models from Section 4.1. If the results are similar, the need to develop a new model for every different VCO circuit topology can be bypassed.

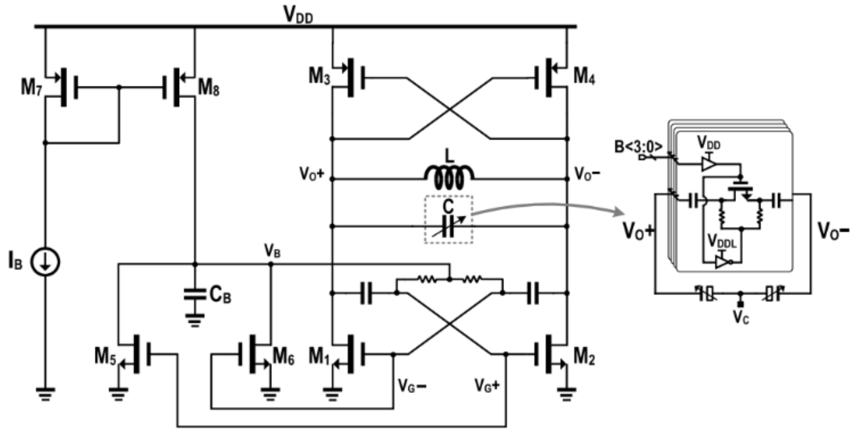

Figure 4.11 – Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO topology. Reprinted from [51]

4.2.1 Dataset analysis

The logic that was used was a resemblance of the one used for the previous circuit, as described in Section 4.1.1, hence a pre-processing was done to the dataset that contained 26108 rows of data, resulting on a dataset of 52216 for the classification ANN and a 469944 for the regression.

4.2.2 Classification results

The tunning of the hyperparameters was made, using the same logic used in Section 4.1.2, and so the study was deliberately omitted, having the final model summarized in the Table 4.17. The evolution of the loss and accuracy of the used model are shown in Figure 4.12 and in Figure 4.14 and the results with the same hyperparameters, used in Section 4.1.3, in Figure 4.13 and Figure 4.15. Additionally, in Table 4.18 the values of loss and accuracy, for both the training and test dataset, between both models, show that results are marginally different. This analysis allows the selection of the first model for both circuits, hence there is no specific need for using different models for both circuits, providing a more generic model ready to be used. Nonetheless, for the purpose of achieving the best results on this work, the second model which was tuned specifically for this circuit was the one chosen.

Table 4.17 – Summary of the Classification Model

Hyperparameter	Value
Input Layer	1 Layer (23 neurons)
Hidden Layers	3 Layers (225,225,200 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	32
Normalization	MinMax (0,1)

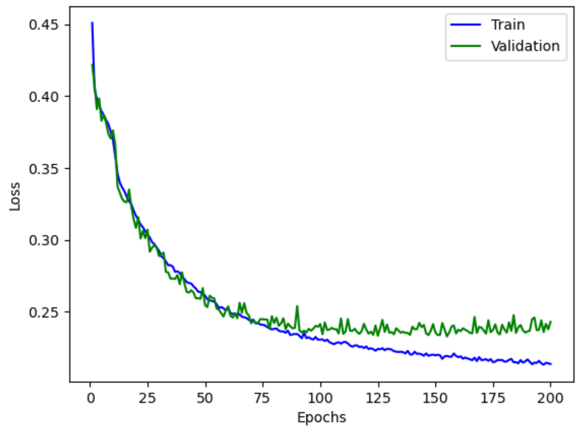

Figure 4.12 - Training and validation loss with model tuned for Class B/C VCO

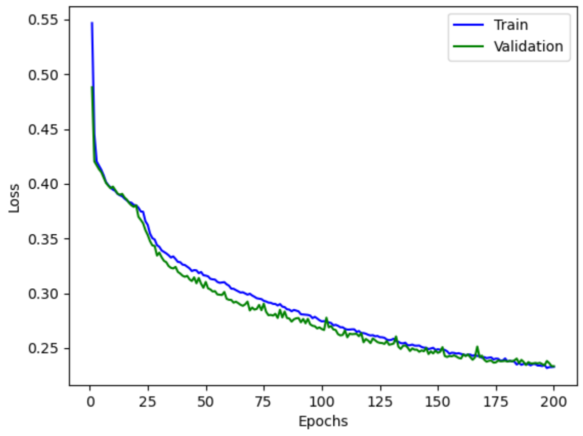

Figure 4.13 - Training and validation loss with Plug and Play from Section 4.1.3

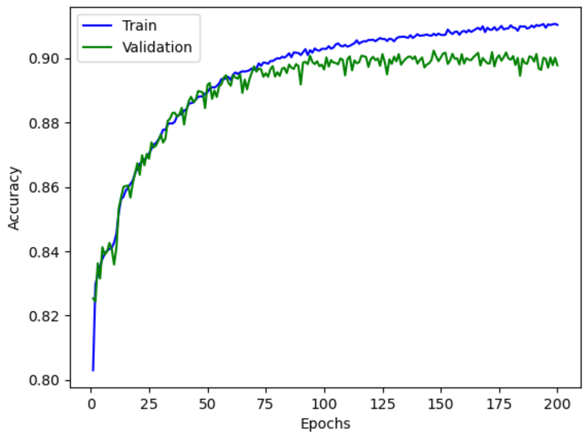

Figure 4.14 - Training and validation accuracy with model tuned for Class B/C VCO

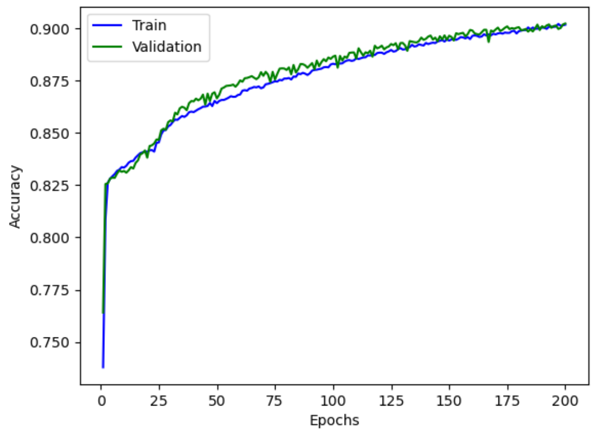

Figure 4.15 - Training and validation accuracy with Plug and Play from Section 4.1.3

Table 4.18 – Training and Test Loss and accuracy between the two models

	Training Loss	Test Loss	Training Binary Accuracy	Test Binary Accuracy
Plug and Play model from Section 4.1.2	2.10e-01	2.35e-01	9.12e-01	9.00e-01
Model tuned for Class B/C VCO	1.87e-01	2.35e-01	9.23e-01	9.01e-01

As aforementioned in the previous circuit, in Section 4.1, in this one it was also used precision and F1 score to ensure the viability of the used model, where its results are shown in the Table 4.19.

Table 4.19 – Train and Test Precision and F1 score Results for model tuned for Class B/C VCO

	Train	Test
Precision	9.29 e-01	9.16 e-01
F1 score	9.50 e-01	9.35 e-01

4.2.3 Regression results

To conclude, the results for the regression ANN are presented, where once more the study of this one was also omitted, since it followed the same logic in Section 4.1.4, where it is possible to observe the model summarized in the Table 4.18, and the evolution of the values of the metrics used for both the training and validation are shown in the Figure 4.16, Figure 4.18, and Figure 4.20, while the ones used in the previous circuit, in Section 4.1.5, in Figure 4.17, Figure 4.19, and Figure 4.21. And in Table 4.21 the values between train and test for both models. As expected, the results prove that the same model used for the first circuit in Section 4.1, could also be used for this one. For this work the best tunned model for this circuit was the one chosen.

Table 4.20 – Resume of the Regression Model

Hyperparameter	Value
Input Layer	1 Layer (24 neurons)
Hidden Layers	3 Layers (200,150,125 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	512
Normalization	MinMax (0.5,1.5)

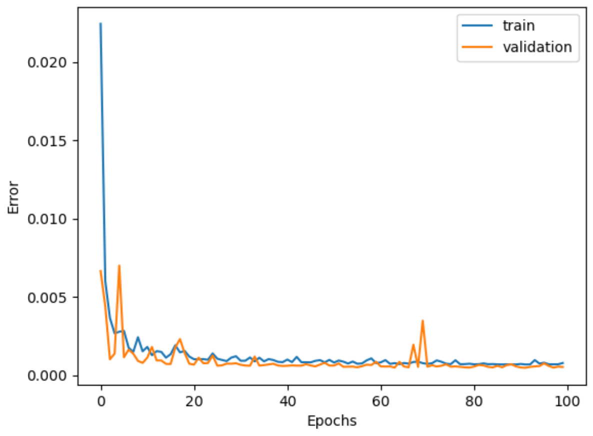

Figure 4.16 - Training and validation MSE with model tuned for Class B/C VCO

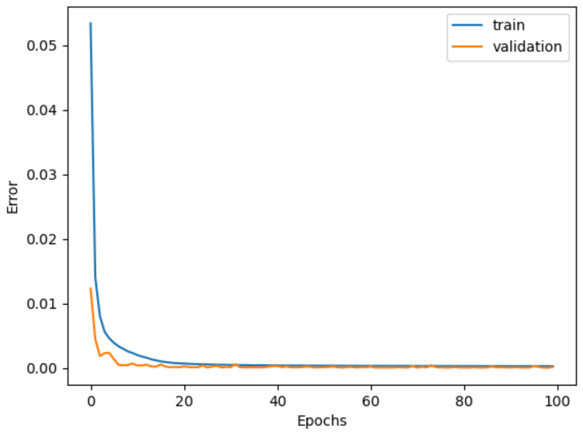

Figure 4.17 - Training and validation MSE with Plug and Play from Section 4.1.5

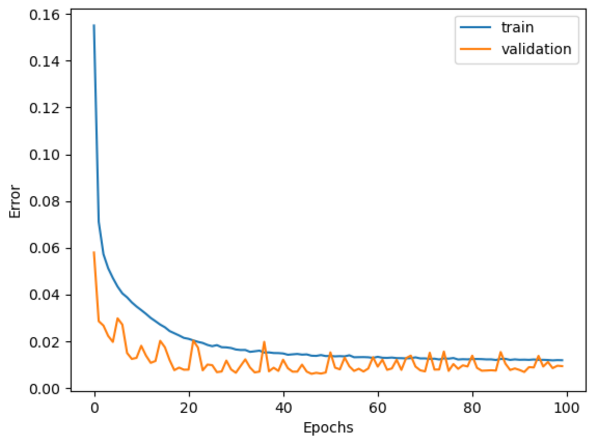

Figure 4.18 - Training and validation MAE with model tuned for Class B/C VCO

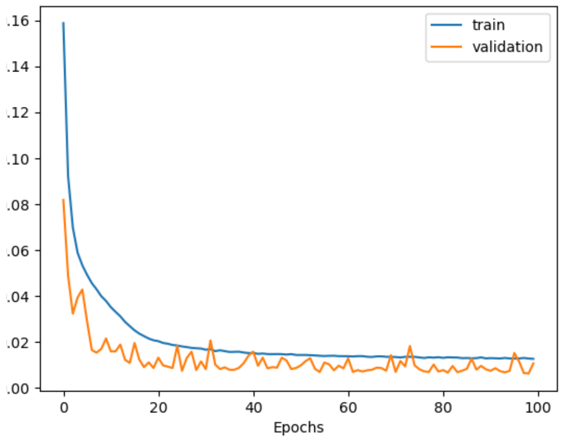

Figure 4.19 - Training and validation MAE with Plug and Play from Section 4.1.5

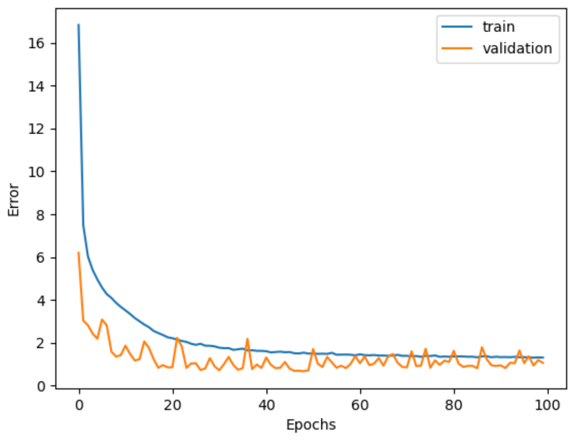

Figure 4.20 - Training and validation MAPE with model tuned for Class B/C VCO

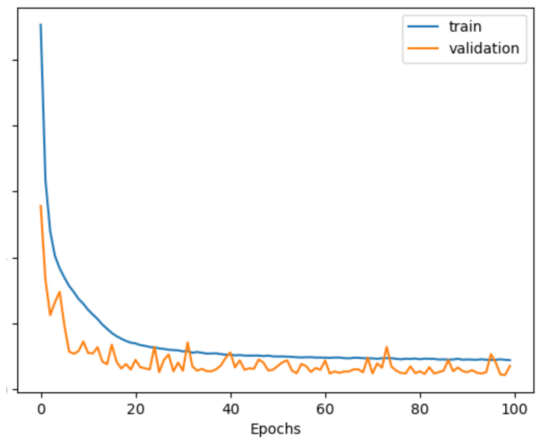

Figure 4.21 - Training and validation MAPE with Plug and Play from Section 4.1.5

Table 4.21 - Training and Test MSE, MAE and MAPE between the two models

	Training MSE	Test MSE	Training MAE	Test MAE	Training MAPE	Test MAPE
Previous Model	2.53E-04	2.38E-04	1.21E-02	1.21E-02	8.44E-01	8.43E-01
Used Model	1.16E-04	1.25E-04	7.75E-03	7.79E-03	8.80E-01	8.87E-01

4.3 Conclusion

It was shown in this chapter the process of training the ANNs and the tuning of the hyperparameters, step by step. Furthermore, the dataset was studied, and its preparation discussed, being this process unique for each ANN. For the circuit, Class-C/D VCO, the results shown are promising, showing good performances in the samples outside the training set, i.e., unseen data. For the circuit, ULP Class-B/C VCO, while the results are satisfactory, they have worst precision and F1 score than the first example regarding the classification, and worst MAE and MAPE regarding the regression. This can be explained, as the dataset for the ULP Class-B/C VCO circuit is considerably smaller when compared with the first circuit, i.e., 26108 and 184230 samples, respectively.

Thanks to the second study, it was proven that the time spent on studying the model for one circuit, does not need to be wasted for a second one, since the model has a high level of generalization, and so is able to present very optimal results when applied to a different VCO. Having all set up, the ANNs were implemented in the design of the circuits.

Chapter 5

AIDA Implementation

This chapter synthetises the procedure and the results of the implementation of the ANNs in the AIDA framework, the tool that carries the optimization of the circuits, as introduced in Section 1.4. This tool, AIDA, is an environment that implements a fully automatic approach, going from a circuit-level specification up to a physical layout description. This work will focus only on the AIDA-C component, which is based on a multi-objective multi-constraint optimization kernel.

The comparison and analysis between the results, obtained with and without the use of ANNs, is performed and discussed. The results of the two circuits previously introduced in Chapter 4 will be carried, starting with the Class-C/D VCO.

5.1 Implementation in AIDA

This section describes all the tests and adjustments that were made in order to obtain the most significant results. To test different approaches, a different threshold, used to determine if the points will be presented to the simulator, was tested and the results analysed. Not only did the threshold came from the usage of the classification ANN, but also from the regression one, as illustrated in Figure 5.1

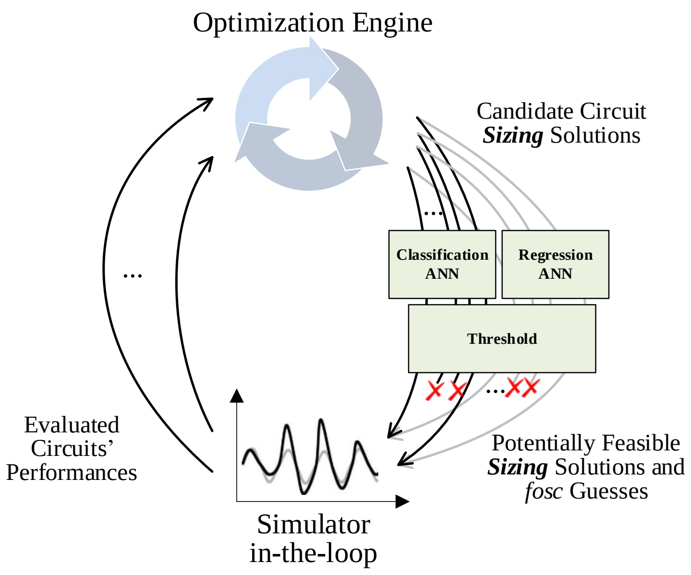

Figure 5.1 – Introduction of Threshold in the Optimization Flow

The classification ANN discards a point (i.e., a candidate sizing solution) based on the value set on the threshold, i.e., if the percentage of corners that are predicted to converge is lower than the value set, then the point is discarded. With the study of the predicted oscillatory frequency, it was possible to conclude that when these values were predicted to be negative, even if the classification predicted many corners to converge, the simulator would not be able to obtain values from that point. Therefore, in case of negative values given by the regression ANN, the point in study will not be sent through the simulator. The functioning of this threshold is illustrated in Figure 5.2, where it is possible to see that thanks to the classification and regression ANN, certain points are filtered and discarded in case of the precited number of corners to converge not meeting the value set up by the threshold or in case the predicted oscillatory frequency is negative.

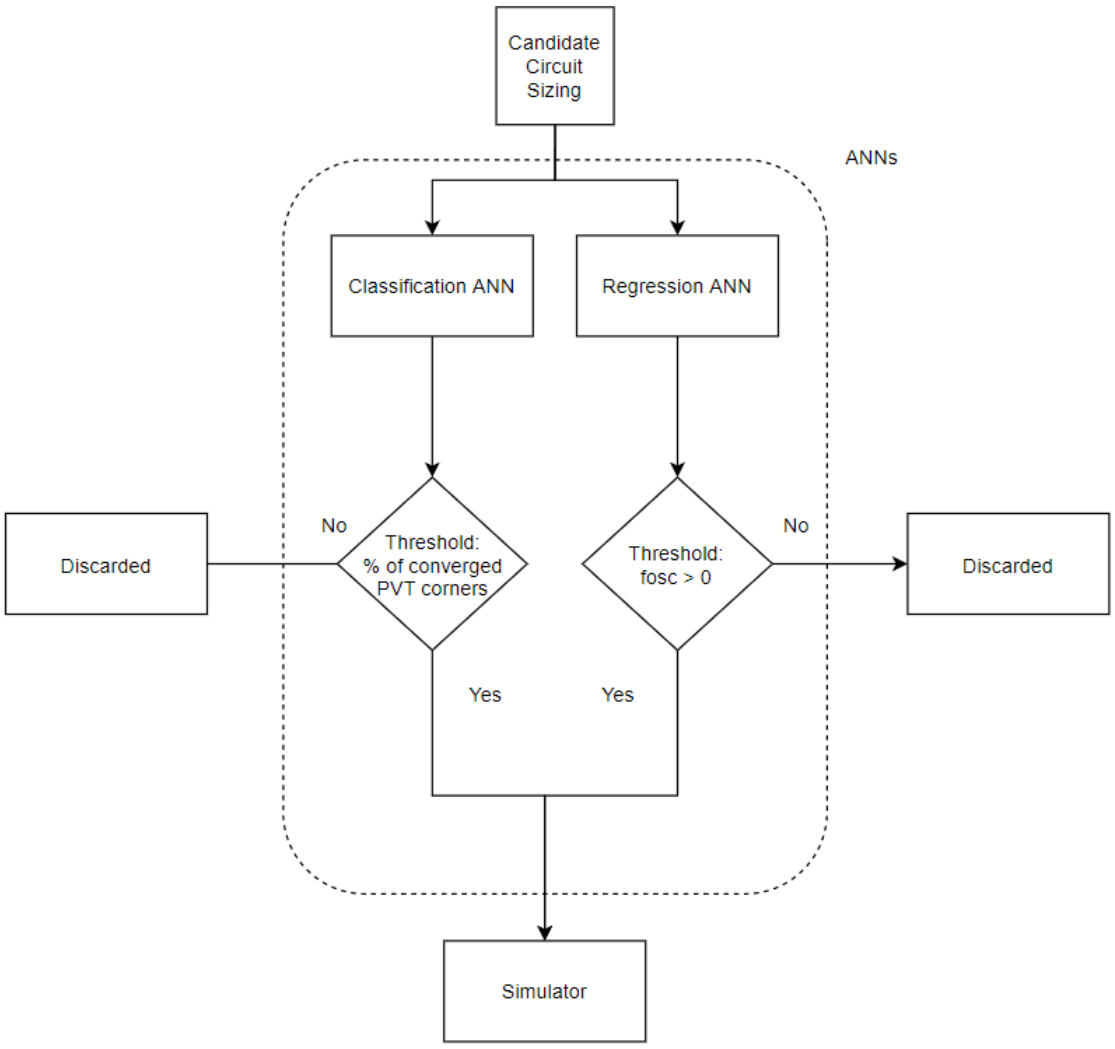

Figure 5.2 – Threshold function in detail

To assess the validation of the results using the ANNs, a reference with the results of the optimization without ANNs, has been established and all the results were challenged against the reference. As a starting point, the original POF obtained was analysed and its points have been presented in raw to the ANNs. The outcome of the ANNs showed that all the points in the POF were also validated with the ANNs, and therefore would be presented to the simulator.

To start the experiments, the threshold was set to 50%, and to make the conditions identical, it was used the same configuration of the optimization without the ANNs, using a population of 256 elements and a total of 150 generations, and the results were obtained with and without the use of the Regression ANN.

5.1.1 50%Threshold

To start off, 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 discarded. With this defined, an optimization was conducted, and the points simulated and discarded computed, arriving to the ratio of points simulated and not simulated illustrated in Figure 5.3. This graph shows that, from the total points that were supposed to be simulated, 18.65% of them were discarded by the ANNs, meaning that almost 20% of the total time required for the optimization was saved with the use of ANNs. Since the optimization without the use of these models lasted 25 days, with the use of these, almost 5 days were economized.

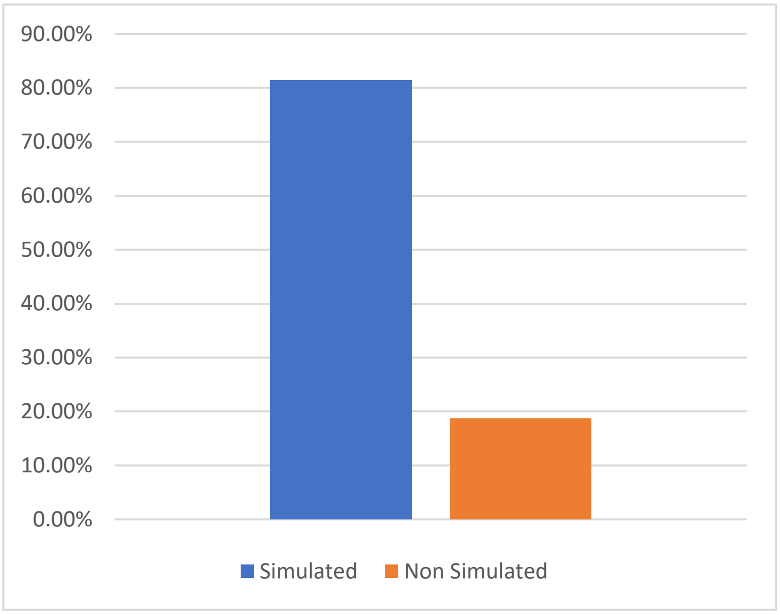

Figure 5.3 – Ratio of points discarded using a value of 50% threshold

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

To start, the results for the first generation were computed, being those presented in Figure 5.4, where the MAPE has 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 given by the simulator.

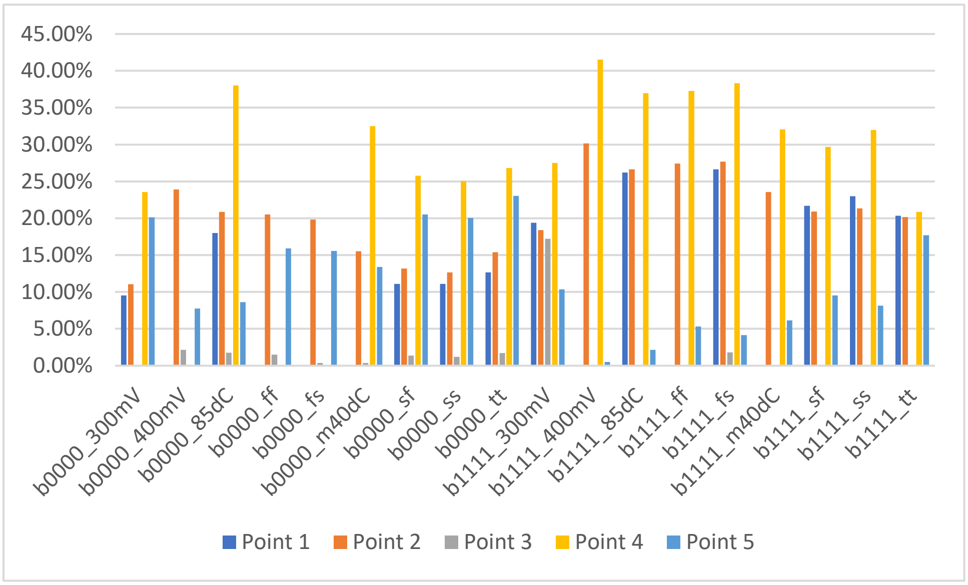

Figure 5.4 – MAPE values from 5 random points of the first generation using a 50% threshold

As the optimization evolve, 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 it is observed in Figure 5.5, where the results were made using the values obtained from the 75th generation.

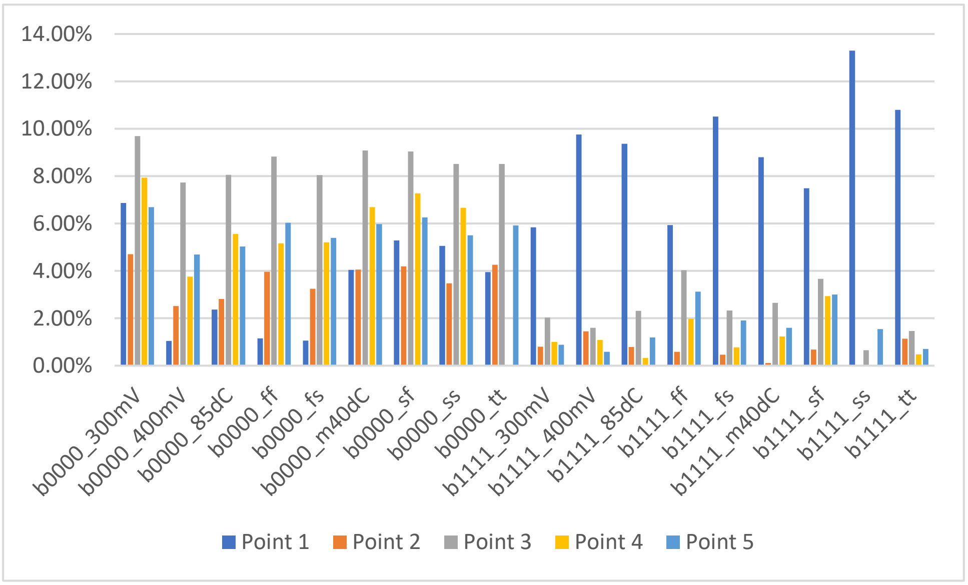

Figure 5.5 - MAPE values from 5 random points of the 75th generation using a 50% threshold

Finally, the results of the last generation, i.e., the 5 random points from the 150th generation show very promising figures, achieving MAPE values lower than 6%, meaning that the regression ANN learns to predict more correctly the oscillatory frequency values as the generations progress, as it is possible to observe in Figure 5.6.

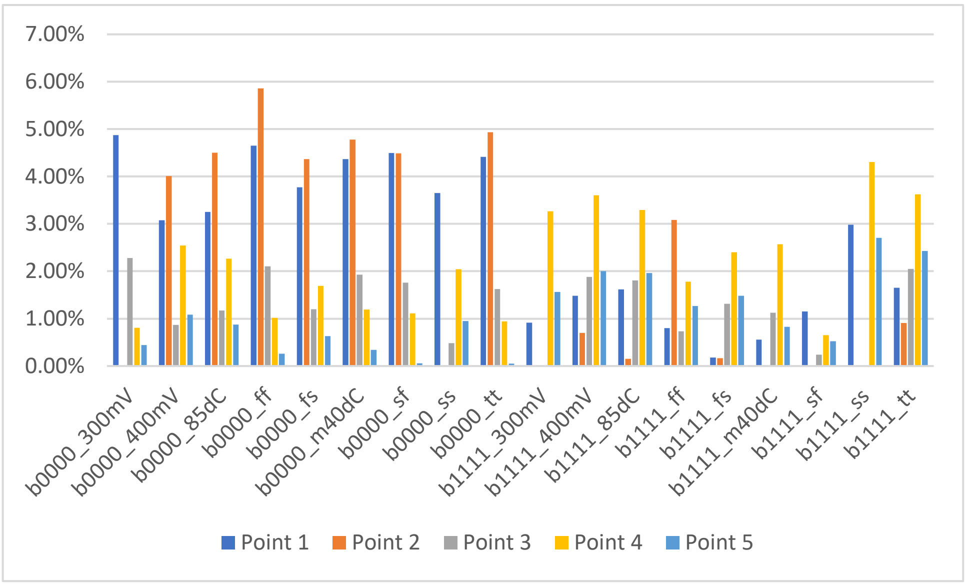

Figure 5.6 - MAPE from 5 random points of the 150th generation using a 50% threshold

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 to values relatively close to the ones given by the simulator. It is also deduced that, along the graphs, the corner SS is the one with the highest error, and the one that the ANN misses most, suggesting that this might be the hardest condition to evaluate the circuit.

5.1.2 Impact of the regression ANN

To prove the value and use of the regression ANN, this one was removed from the implementation, and another optimization with the same characteristics, a population of 256 elements that were optimized for 150 generations, was conducted. This led to a discard of almost 30% of the total points that would be simulated, as 28.23% of the total points were discarded, which was higher than the previous results, as the Figure 5.7 shows. This increase in points discarded can be explained since, without the regression ANN, the simulator was not able to come up with the best oscillatory frequencies, and so the points that progressed during the evolution of the generations, were not so optimized, hence more solutions were discarded by the classification ANN.

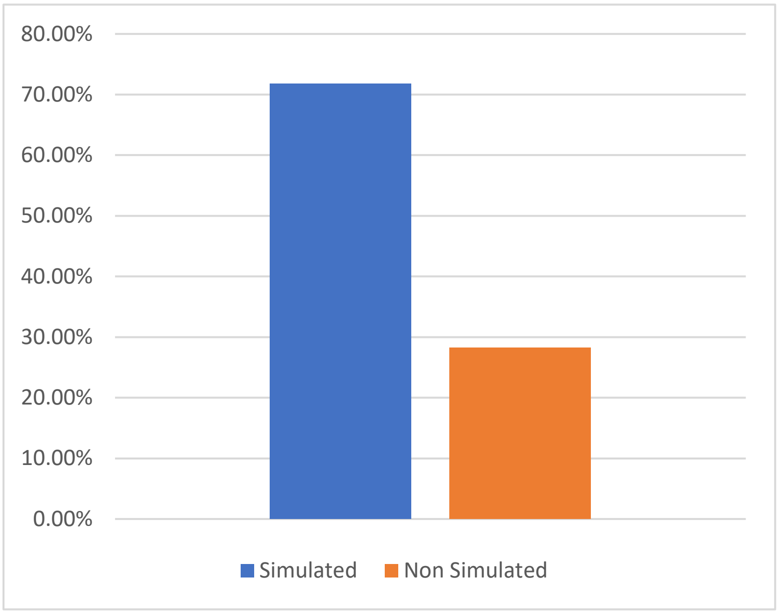

Figure 5.7 – Ratio of points discarded without the use of the regression ANN

The optimal points for each optimization, with and without the use of the regression ANN, were retrieved and a POF was obtained. These two POFs were then compared against the POF obtained without the use of the ANNs, considered the reference, as depicted in Figure 5.8. As the results show, the use of the regression ANN proves to be valuable since that, even though its corresponding POF shows that the solutions obtained have better results in terms of phase noise, however, in terms of power the results are considerably worse, never achieving values lower than 1.30E-03. Where with the use of the regression, the POF obtained is very similar to the reference one. Some solutions have worse values of power, however there are some that achieve values lower than 9.00E-04, and better results in terms of phase noise, reaching values of -136.00. Therefore, the use of the regression ANN proves to be essential, as it helps the simulator to converge thanks to the predicted value of the oscillatory frequency.

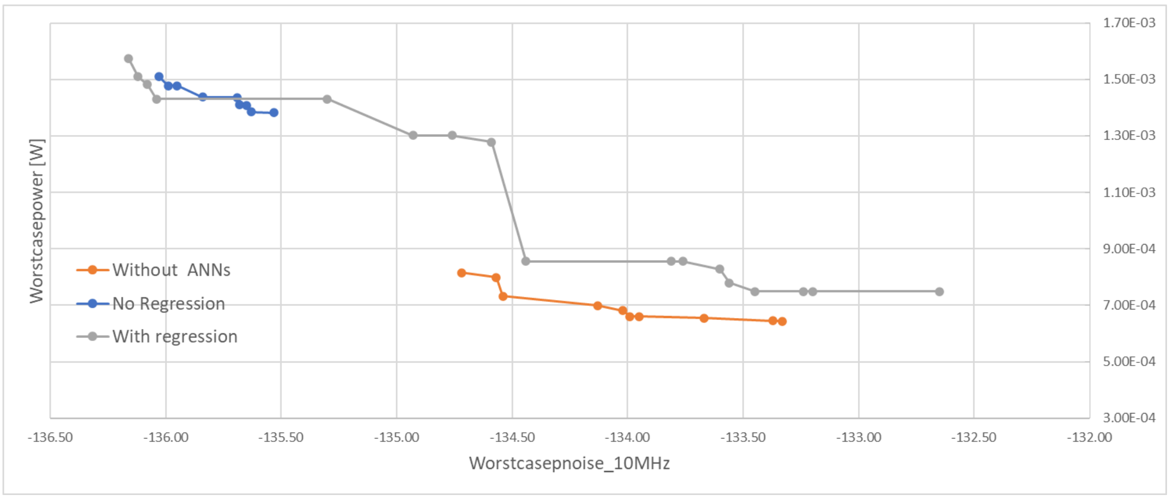

Figure 5.8 – Comparison of POFs obtained with a value of 50% threshold and with and without the regression ANN versus the original

5.1.3 75%Threshold

Next, the threshold was defined to 75%, being this value more rigid and so it is expected that the number of points discarded is higher. The same configuration for the optimization was used, and so the points simulated were computed, reaching to the ratio showcased in Figure 5.9. As expected, the efficiency of the classification ANN was higher, as the points discarded were higher than the results before, reaching a value of 19.65% of points discarded from the total points that would be fed to the simulator.

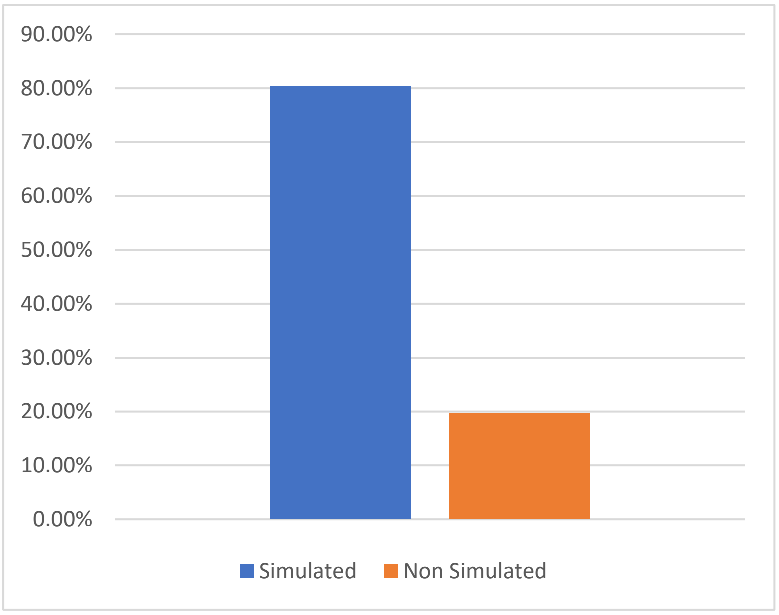

Figure 5.9 - Ratio of points discarded using a value of 75% threshold

To verify the results, the MAPE values along the evolution of the generations was measured, being presented on Figure 5.10, Figure 5.11, and Figure 5.12 the results for the first, 75th and 150th generation, respectively.

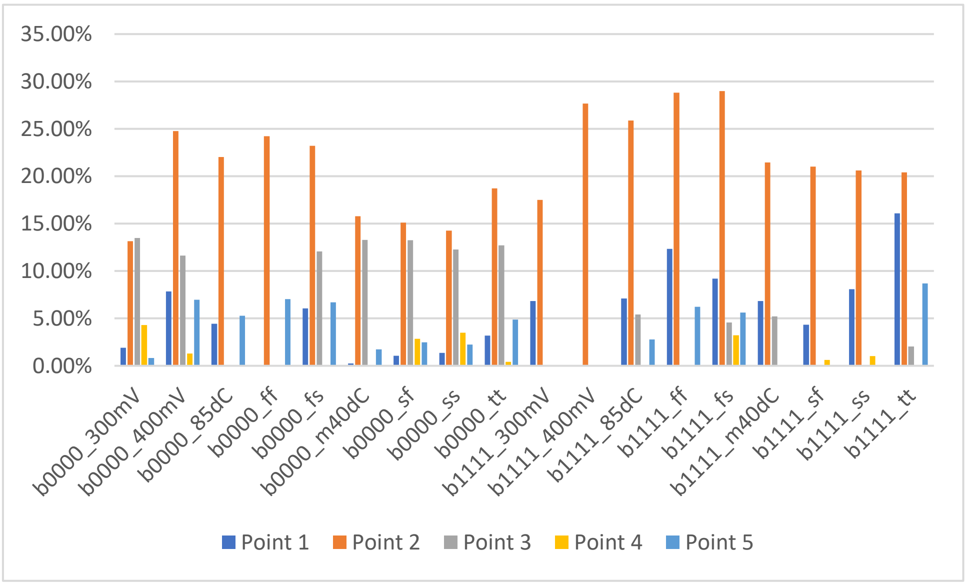

Figure 5.10 – MAPE values from 5 random points of the first generation using a 75% threshold

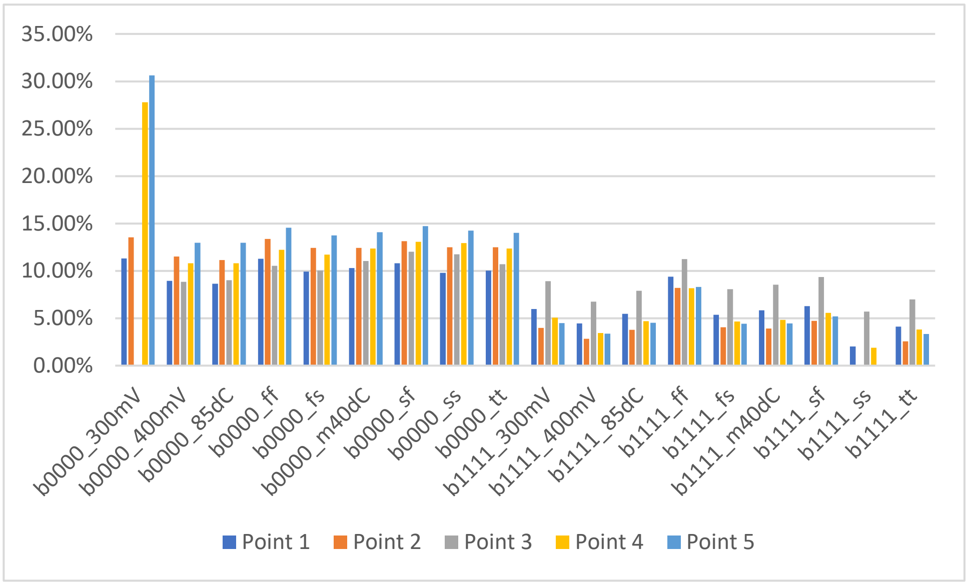

Figure 5.11 – MAPE values from 5 random points of the 75th generation using a 75% threshold

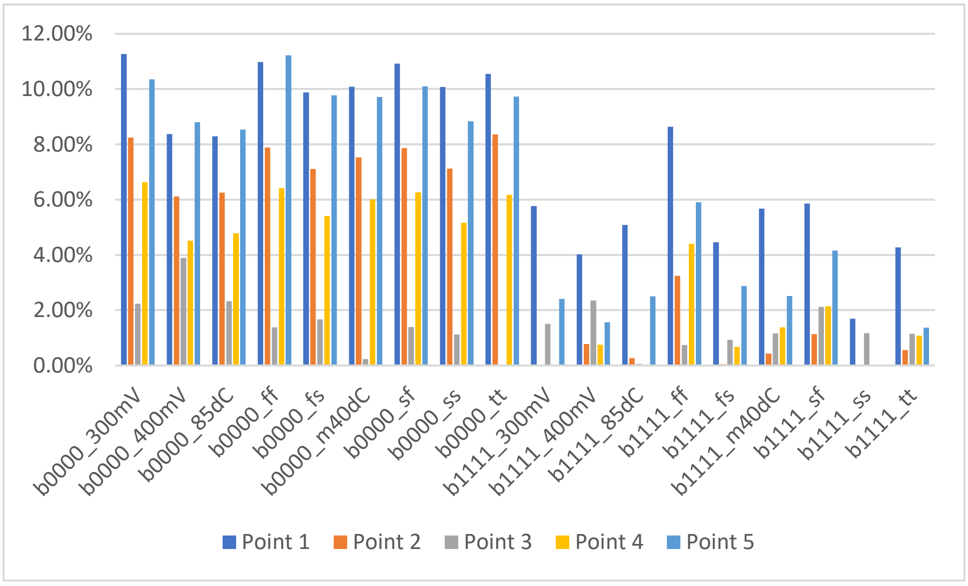

Figure 5.12 – MAPE values from 5 random points of the 150th generation using a 75% threshold

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 optimal as the ones in the Section 5.1.1, the regression ANN is able to efficiently predict the oscillatory frequency, even with a higher value of threshold. To conclude, the resulting POF was extracted and again compared with the reference one as illustrated in Figure 5.13. The results show that the points obtained were again very similar to the references one, even reaching values of power and phase noise better than the reference one. In terms of power, all the values are lower than 7.91E-04, and lower than -134.19 in terms of phase noise.

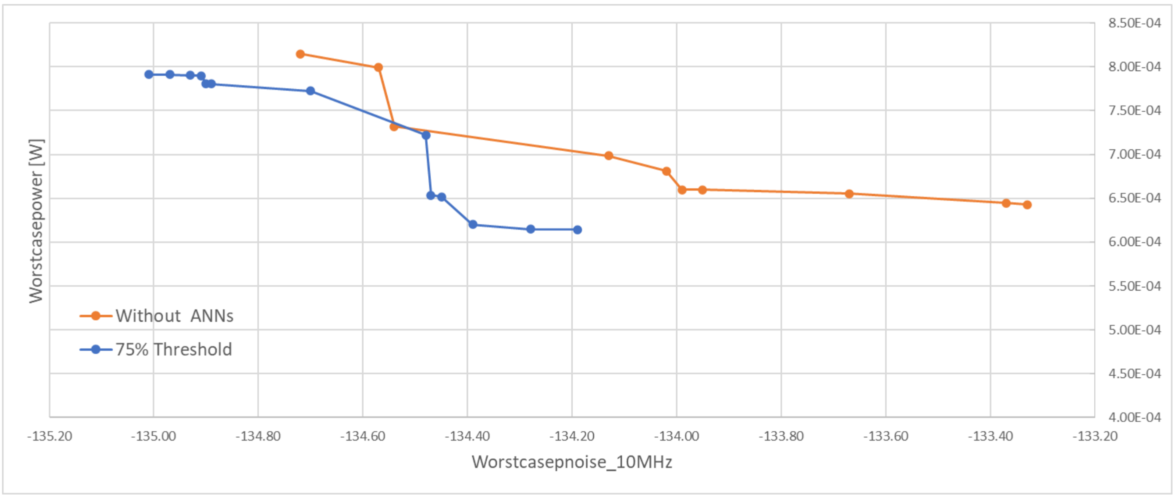

Figure 5.13 – POF obtained with 75% threshold

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 perform optimizations in less time, without compromising the results. Next the threshold is set up with an even higher and more eager value, to achieve better results, as the value is set to 90% and 100%.

5.1.4 90 and 100% Threshold

After setting the threshold with a higher value, the same logic and optimization configuration was used as before, and so the number of points that were discarded, from the total number, were extracted, and as the results in Figure 5.14 show, the number were lower than the ones obtained from use of a 75% threshold.

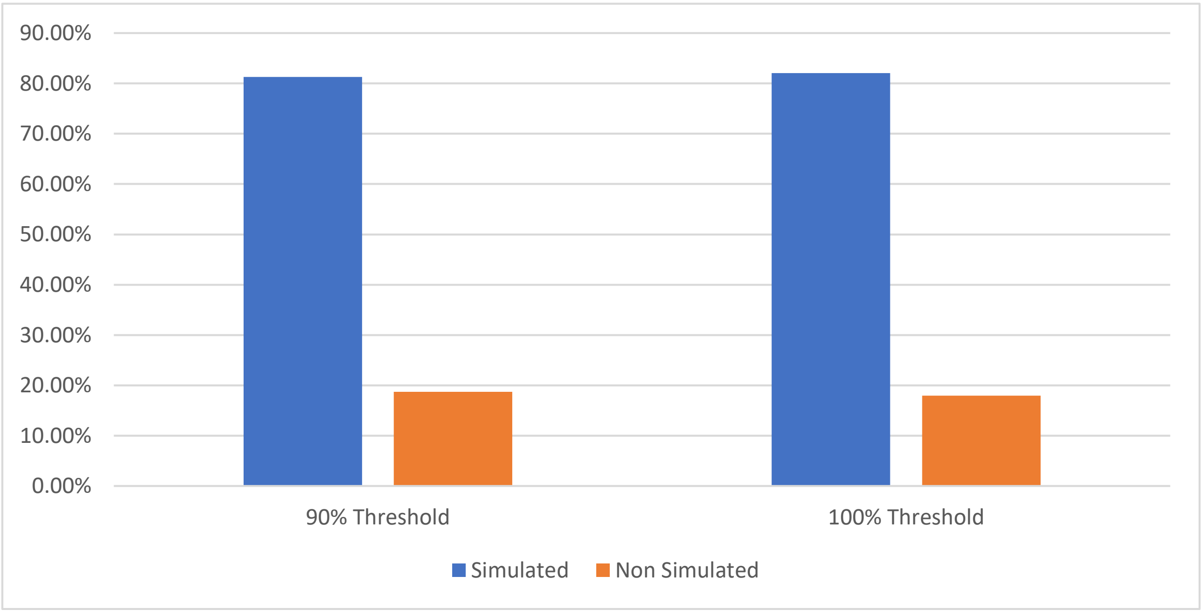

Figure 5.14 – Ratio of points discarded with 90% and 100% threshold

From the inspection of the POFs obtained in Figure 5.15 and Figure 5.16, it is possible to conclude that the results were worse, as the POFs obtained are further away from the reference one. With a 90% value of threshold, some points have good values of phase noise, however the values of power are too high, whereas with a 100% the values of power and phase noise are considerably worse.

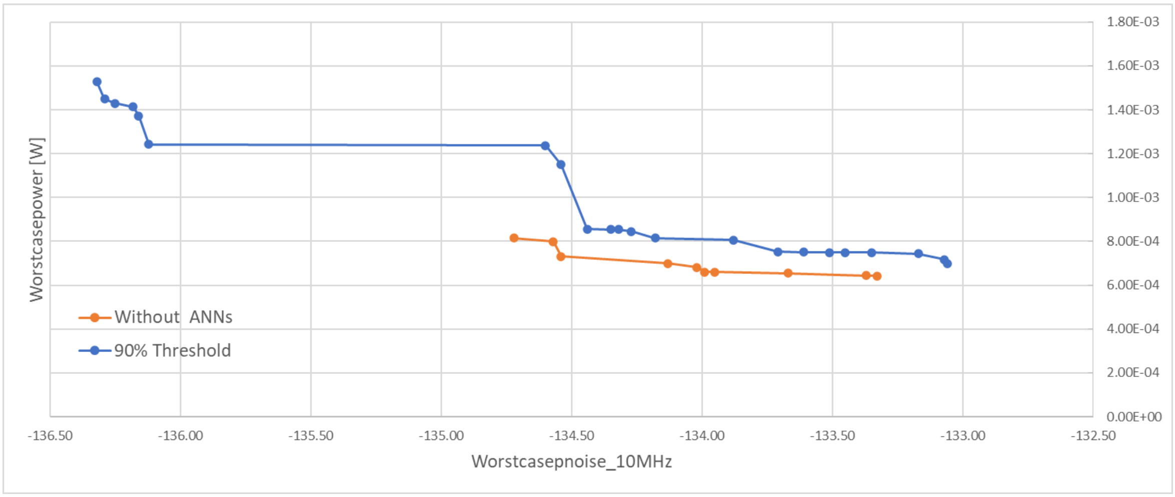

Figure 5.15 - POF obtained with 90% threshold

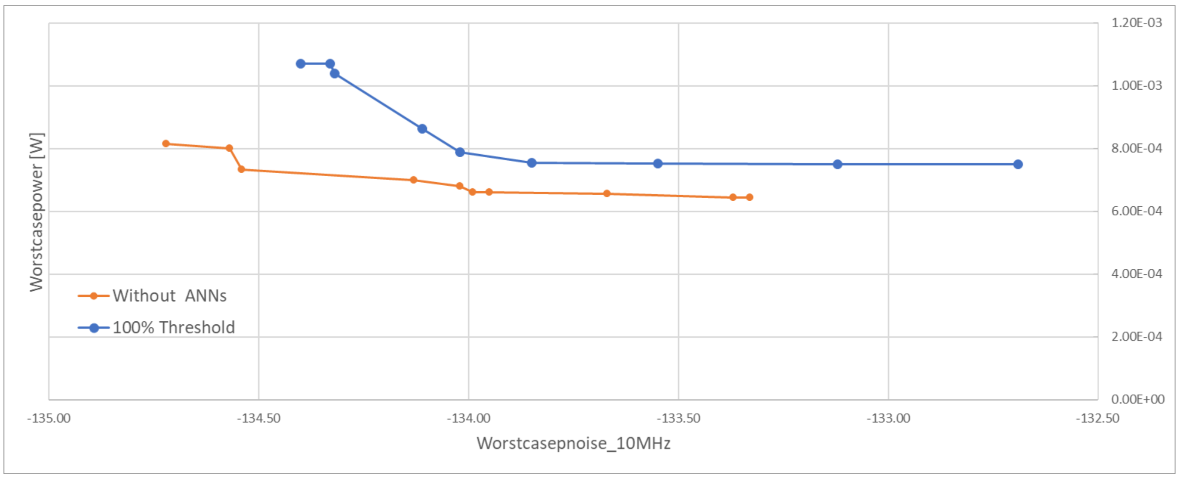

Figure 5.16 - POF obtained with 100% threshold

Therefore, not only were the number of points discarded lower, but also the values of the optimization worse, which led to the conclusion that the value set to the threshold was too high, and so it wasn’t favourable to use high values. Having tested different values for the threshold, the best results arrived from the value of 75%, and so this was the value used for the following tests.

5.1.5 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, and to understand if there was some PVT corner which was harder to predict or evaluate. Hence, for an optimization produced by a value of 75% threshold, points that were sent to the simulator were analysed along the evolution of the generations.

Starting with the ones obtained from the first generation, there were 15 points that 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 they would fail to converge in one specific corner, for each point. These would miss two times for the ff corner, one in the first tunning mode, one in the second, and one time in the ss corner in the second tunning mode. However, the simulator did not manage to obtain values from more than only those three corners, and on more occasions, as the Figure 5.17 shows the frequency of the non-simulated PVT corners.

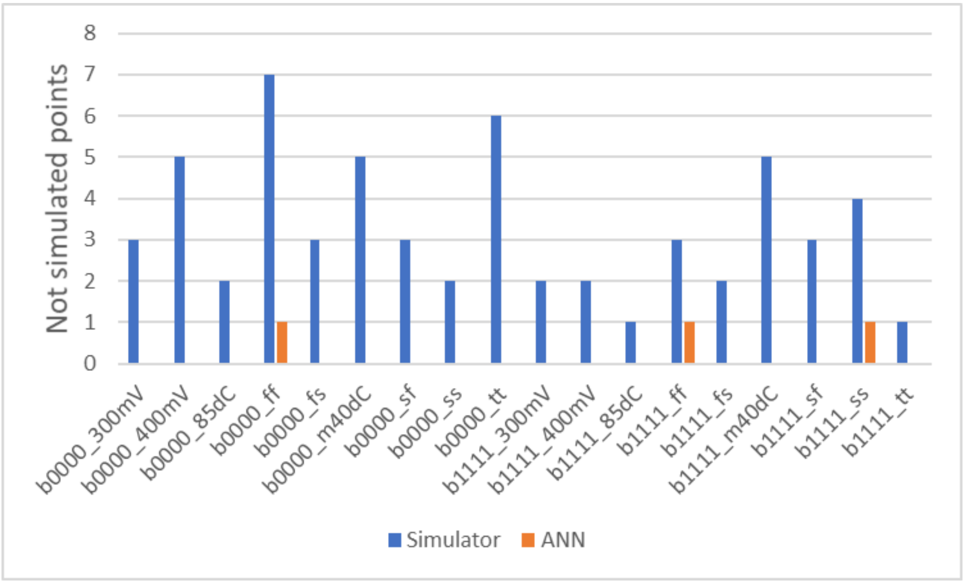

Figure 5.17 – Difference between the prediction of the classification ANN and the output of the simulator for the first generation

Advancing to the 75th generation, there were fewer differences in the output of the ANNs and the simulator. Analysing the 23 points that the ANNs fed to the simulator, this one predicted that one point would fail in the ss corner for the second tunning mode, where the simulator failed to simulate for three corners, and in one more occasion for the corner that the ANNs predicted, as the Figure 5.18 shows.

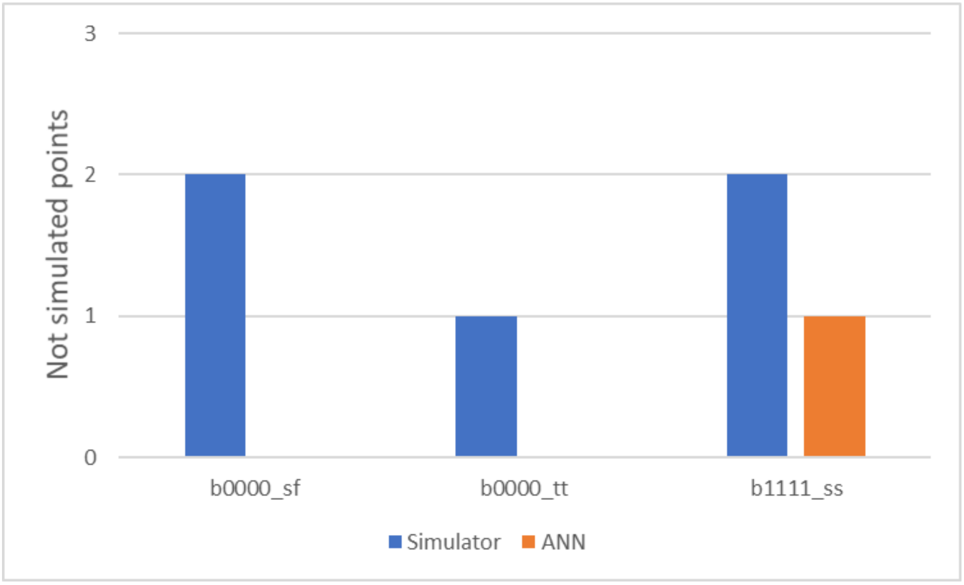

Figure 5.18 - Difference between the prediction of the classification ANN and the output of the simulator for the 75th generation

Finally, in the 150th generation, the ANNs predicted that 24 points should be simulated, with zero cases of corners not converging, where the simulator failed to obtain values in seven of those points, and in seven different corners. The ss corner in the second tuning mode was the one which missed the most cases, as observed in Figure 5.19.

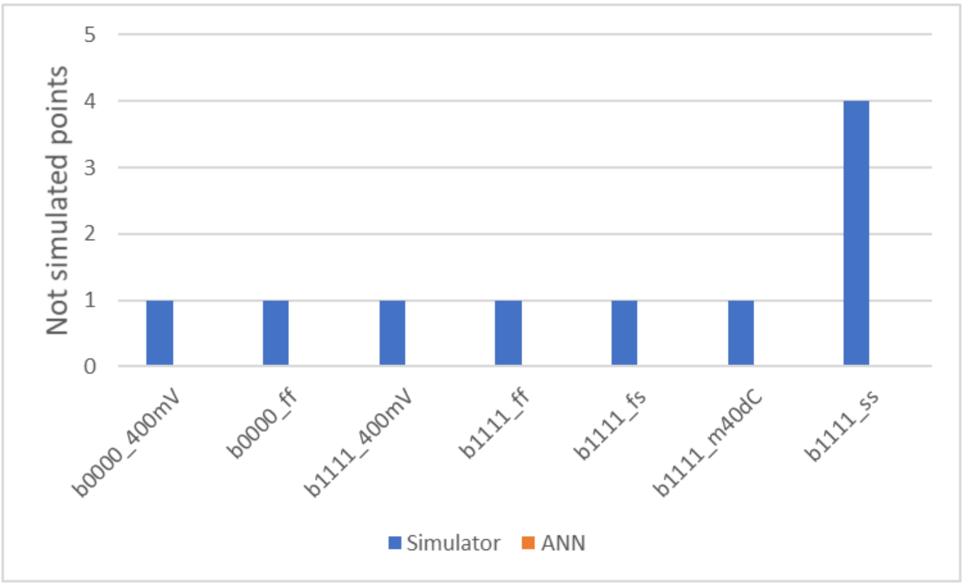

Figure 5.19 - Difference between the prediction of the classification ANN and the output of the simulator for the 150th generation

From an earlier analysis of the dataset, it was possible to obtain the number of points that converged for each PVT corner. The Table 5.1 showcase these results, and it can be concluded that the corner ss, for both tunning modes, is the one with the highest number of *NaN*, meaning that this one is the most difficult corner to retrieve the performance values. Additionally, in the second tunning mode this happens more often, reaching to almost half of the cases. This explains why the ANNs predict that corner to fail more often. Furthermore, it also explains why the simulator shows to have more difficulties obtaining values from that corner, since this corner presents to be the hardest to get values from.

Table 5-1 – Percentage of converged points for each corner

	tt	ff	fs	sf	ss	300mV	400mV	m40dC	85dC
b0000	87462	87200	87881	88407	79130	82206	89449	87094	86678
%Total	94.95%	94.66%	95.40%	95.97%	85.90%	89.24%	97.11%	94.55%	94.10%
b1111	85143	84631	86717	82049	52680	76735	89021	82377	87513
%Total	92.43%	91.88%	94.14%	89.07%	57.19%	83.30%	96.64%	89.43%	95.00%

The results led up to the conclusion that, the points that the ANNs predict that should be simulated, in some cases, the simulator gives different outputs, however, this tends to happen less often as the generations progress. Additionally, the most difficult corner to obtain values is the ss corner in the second tuning mode, which explains why the ANNs fails more often in those cases, and why the MAPE values were higher in those corners.

5.1.6 New Specifications

To prove the efficiency of the ANNs with different changes within the same circuit, a new test was made, and the same ANNs that were trained for the previous specifications, were faced upon a new setup where the specifications for the circuit were slightly different. For this setup, the circuit was meant to operate at 2.4GHz, having a range of frequency from 2.3GHz to 2.5GHz, and the constraints of the phase noise were changed, being the new ones tighter in 5dBc/Hz.

Having all prepared, a new optimization was ran, using a population of 256 elements for 200 generations, and the ratio of the points simulated and discarded was produced as Figure 5.20 illustrates. The number of points discarded was low, having a value of only 9.51%, which can indicate immediately that the efficiency of the ANNs were poorer.

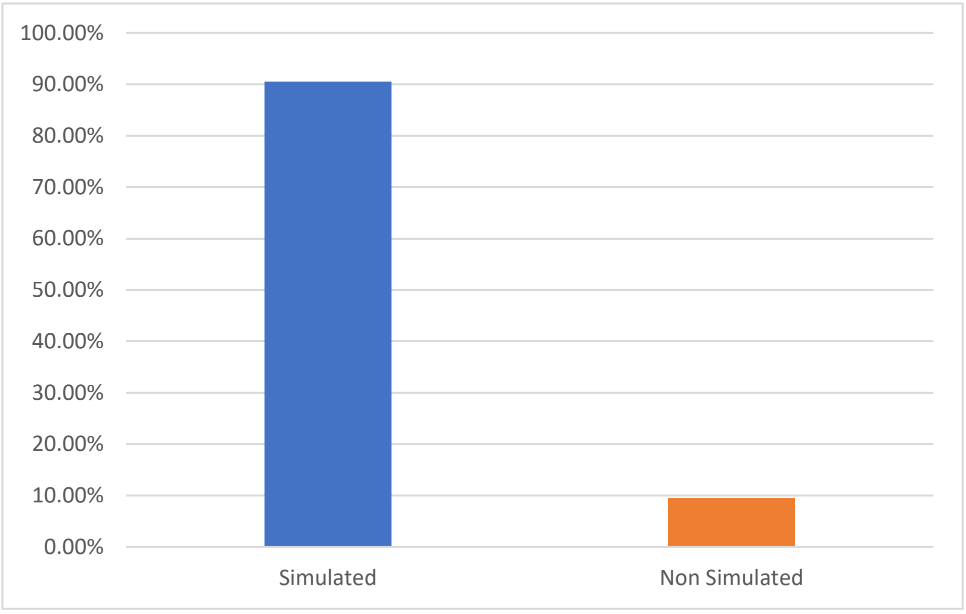

Figure 5.20 – Ratio of points discarded for the new specifications

Even with a low number of points discarded, the optimal points obtained could present good values, and so the POF was retrieved and compared with the one obtained without the use of the ANNs. Analysing the results, one can observe that the POF obtained while using the ANNs has better results in term of phase noise, as the solutions are always lower than -138.50. However, that's not the case when looking at power. The values are higher, achieving results that don't get lower than 6.00E-04, while the ones without using the ANNs, are lower, where the values of power never reach 5.00E-04, as observed in Figure 5.21.

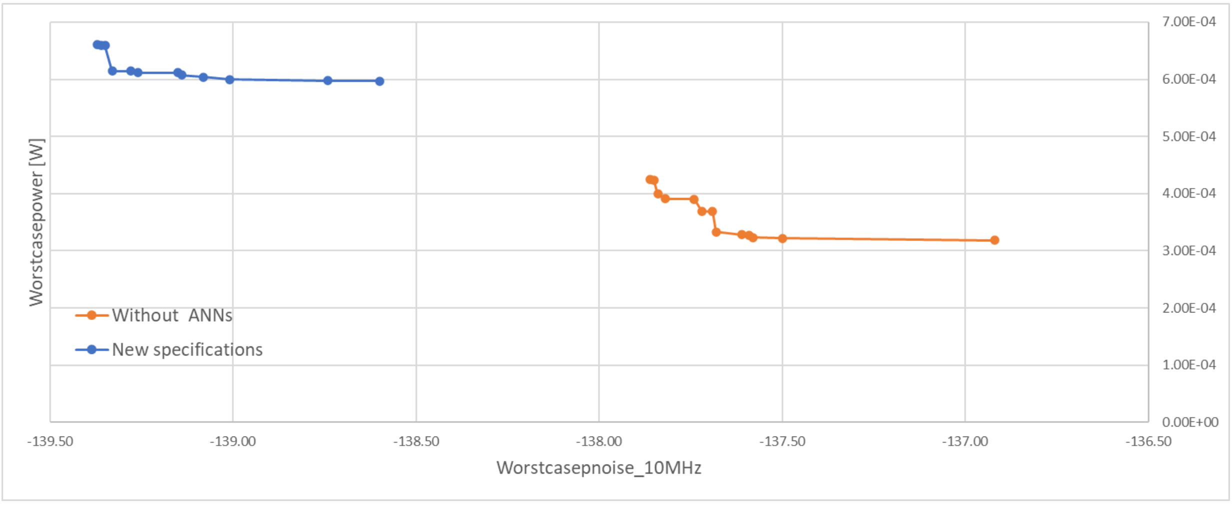

Figure 5.21 – POF obtained with the new specifications

Ultimately, the results are acceptable, with a reduction of almost 10% of the time, which corresponds to a reduction of almost 3 days since the optimization took about 26 days to complete without the use of ANNs. Even though, the POFs are slightly distanced, the optimal points retrieved proved to be better regarding the phase noise, with slight inferior results in power. This can lead to the deduction that, the ANNs do perform well when facing different specifications, and so, is possible to use already trained ANNs for different configurations for the circuit that they were trained for.

5.2 Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO Results

Finally, the ANNs were implemented in the second circuit, the Class-B/C VCO. The same approach was used as the one used in Section 5.1, and so it was performed an optimization and the number of points that were simulated were extracted. For this circuit, the configurations were changed, as the ones used without using the ANNs were also modified, and so, a generation of 256 points was used, and it was performed an optimization for 97 generations. As Figure 5.22 shows, from the total points that were simulated, 17.27% were discarded, being this number close to the 19.65% that was obtained for the first circuit, the Class-C/D VCO, with a value of also 75% threshold. Since the optimization without using ANNs took 25 days, by using these ones, more than 4 days were economized.

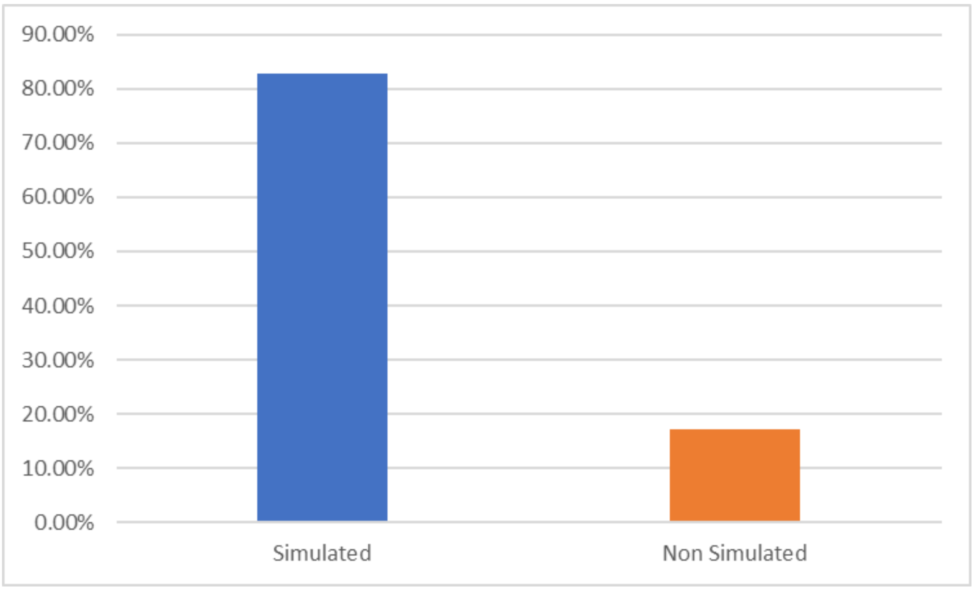

Figure 5.22 - Ratio of points discarded for the new circuit

To prove the validity of the results, the POF was obtained, and it was compared against the one without the use of ANNs, named reference, as showed in Figure 5.23 .The results show to be very promising, as the values of power obtained are lower, being close 1.50E-04, as the ones obtained without using the ANNs are more close to 2.00E-04. Regarding phase noise, the results are very similar, even recording three points that reach values lower than –130.55, whereas in the reference POF the results never get lower than this value.

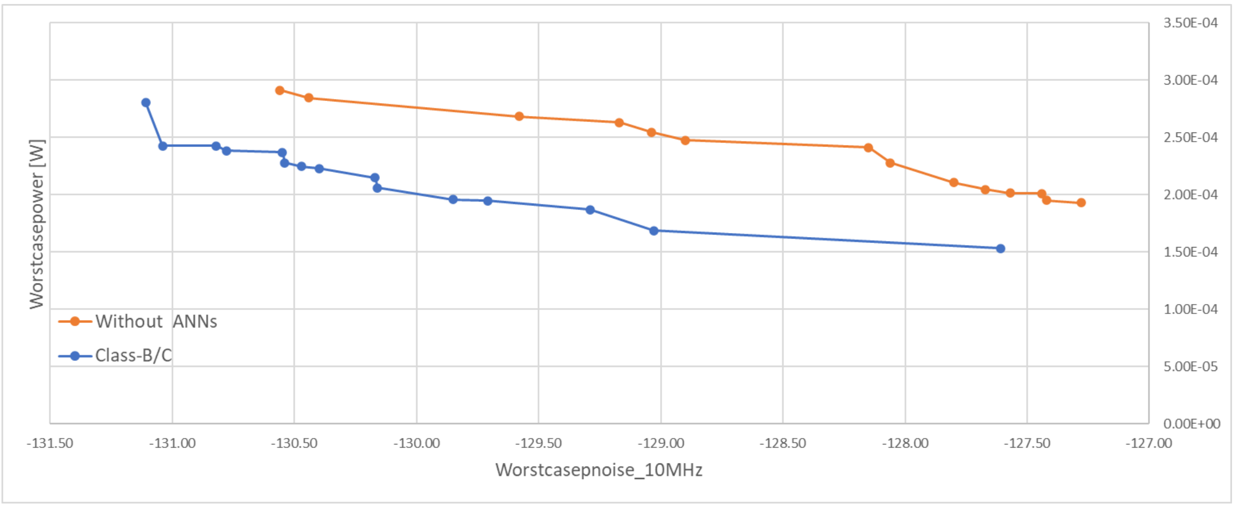

Figure 5.23 - POF obtained for the Class-B/C

Chapter 6

Conclusions

This chapter exposes not only the conclusions of the work done for this dissertation, but also the aim of the project for further optimizations and use of ANNs applied to analog IC sizing.

The work here exhibited, showed an approach to optimize the sizing of analog IC circuits with the help of two ANNs, one classification and one regression. They both proved to be essential, as the use of the classification and regression ANN discard unwanted solutions, and the use of the last one makes the results even more accurate and approximate to the ones obtained at the first place, therefore reducing the whole optimization process.

Two circuits, with the same constraints, were studied, and its design optimized.

The first one was a Class-C/D, where thanks to the ANNs, it was possible to reduce almost 20% of the optimization process, without compromising the results. The optimal points obtained were very similar, even better in some cases, to the ones retrieved without using the ANNs. For the same circuit, the ANNs also proved to work for different specifications, however with less efficiency, as the time saved was close to 10%, and the solutions obtained better in some cases, but worse in other.

A second circuit, an Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO, was analysed and optimized, having the intention to use the same ANNs used for the previous circuit, the Class-C/D. To do so, a careful study was again made, using the same logic used for the previous circuit, and the new models obtained were discussed and compared with the previous ones. The results were marginally different, proving that, there is no need to spend time training new ANNs, and so emphasizing the generalization of these ones. The design of this circuit, was successful, reducing almost as much time as for the first circuit, and its optimal points showed very promising results.

To sum up, the results proved that the ANNs not only show to be very useful when designing a circuit, but also, for a different one, of the same type, without having the need to do an additional train of the ANNs. However, the results for a different specification fall somehow short of the ones expected, as the model implemented is not able to achieve its best results.

6.1 Future work

This dissertation proved that the use of ANNs reduce the time to design an analog IC, even if the effort required for their training must be taken in consideration. Having this thought in mind, it is possible to take a step further.

First of all, even though the ANNs prove to have a high level of generalization towards different types of VCOs, there is still required time to train these ones. A future objective would be to reduce this time, which would be possible with additional computation resources. In this dissertation, the training of the ANNs, was performed with human interaction, where each hyperparameter was optimized manually.

On a future perspective, this process could be optimized by using algorithms finding the best values for the hyperparameters, resulting on an even further reduction in time.

On a second approach, the training of this ANNs was executed using a dataset that was given, so this one was already known and well defined. However, with the ANN architecture well defined, the implementation of the models can be used without the need for the complete dataset at the beginning, and therefore the train of the ANNs would not be done before, but as part and during the optimization process. This would work in a dynamic way, such that the ANNs could be implemented in the optimization loop, and they would be trained during its process, resulting at first, on using only the normal procedure, without the ANNs, and as soon as the accuracy of the ANNs would be high enough, they would start to be used and therefore to reduce the optimization time.

One challenge that comes with this approach, is the flexibility that the ANNs would have to show when facing small samples of data and be trained fast and accurate enough for the process to be worth its implementation.

References

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

[2]. The Business Research Company, “Integrated Circuits Global Market Report 2020” tech. rep., 2020.

[3]. G. G. E. Gielen and R. A. Rutenbar, “Computer-aided design of analog and mixed-signal integrated circuits,” in Proceedings of the IEEE, vol. 88, no. 12, pp. 1825-1854, Dec. 2000.

[4]. H. Graeb, “ITRS 2011 Analog EDA Challenges and Approaches,” *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, 2012, pp. 1150-1155.

[5]. G. Gielen and R. Rutenbar, “Computer-aided design of analog and mixed-signal integratedcircuits, “Proceedings of the IEEE,vol. 88, no. 12, pp. 1825-1854, December 2000.

[6]. G. Gielen, “CAD tools for embedded analogue circuits in mixed-signal integrated systemson chip,” in IEE Proceedings on Computers and Digital Techniques, vol. 152, no. 3, pp.317–332, May 2005.

[7]. G. G. E. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," in *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825-1854, Dec. 2000.

[8]. Bi, Yu & Yuan, Jiann & Jin, Yier. (2015). Beyond the Interconnections: Split Manufacturing in RF Designs. Electronics. 4. 541-564. 10.3390/electronics4030541.

[9]. R. Martins et al., “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,” in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 11, pp. 3965-3977, Nov. 2020.

[10]. R. Martins, N. Lourenço, N. Horta, J. Yin, P. Mak and R. P. Martins, “Many-Objective Sizing Optimization of a Class-C/D VCO for Ultralow-Power IoT and Ultralow-Phase-Noise Cellular Applications,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 1, pp. 69-82, Jan. 2019.

[11]. "Mentor Graphics," [Online]. Available: http://www.mentor.com.

[12]. Lourenço, Nuno & Martins, Ricardo & Canelas, António & Póvoa, Ricardo & Horta, Nuno. (2016). AIDA: Layout-Aware Analog Circuit-Level Sizing with In-Loop Layout Generation. Integration, the VLSI Journal. 55. 10.1016/j.vlsi.2016.04.009.

[13]. M. G. R. Degrauwe et al., “IDAC: an interactive design tool for analog CMOS circuits,” in IEEE Journal of Solid-State Circuits, vol. 22, no. 6, pp. 1106-1116, Dec. 1987.

[14]. F. El-Turky and E. E. Perry, “BLADES: an artificial intelligence approach to analog circuit design,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 8, no. 6, pp. 680-692, June 1989.

[15]. 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, Heidelberg. https://doi.org/10.1007/978-3-642-36329-0_7.

[16]. H. Koh, C. Sauin, P. Gray, “OPASYN: a compiler for CMOS operational amplifiers,”IEEE Trans. CAD, Vol. 9, No. 2, pp. 113-125, February 1990.

[17]. G. Jusuf, P. Gray, A. Sangiovanni-Vincentelli, “CADICS - Cyclic ana- log-to- digital converter synthesis,” Proc. ACMNEEEE ICCAD, pp. 286- 289, 1990.

[18]. G. Gielen, H. Walscharts, W. Sansen, “ISAAC: a symbolic simulator for analog integrated circuits,” IEEE JSSC, Vol. 24, No. 6, pp, 1587-1597, Dec. 1989.

[19]. K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002.

[20]. B. Liu, N. Deferm, D. Zhao, P. Reyaert, G. Gielen, “An Efficient High-Frequency Linear RF Amplifier Synthesis Method Based on Evolutionary Computation and Machine Learning Techniques,” IEEE TCAD of Integrated Circuits and Systems, vol. 31, no. 7, pp. 981-993, July 2012.

[21]. R. Póvoa, et al., “LC-VCO automatic synthesis using multiobjective evolutionary techniques,” IEEE International Symposium on Circuits and Systems, pp. 293-296, Jun. 2014.

[22]. E. Afacan and G. Dündar, “A mixed domain sizing approach for RF Circuit Synthesis,” in IEEE Inter. Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 1–4, Jun. 2016.

[23]. R. González-Echevarría, et al., “An automated design methodology of RF circuits by using Pareto-optimal fronts of EM-simulated inductors,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 1, pp. 15-26, Jan. 2017.

[24]. E. Afacan and G. Dündar, “Design Space Exploration of CMOS Cross-Coupled LC Oscillators via RF Circuit Synthesis,” 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, pp. 1-4, July 2018.

[25]. E. Afacan and G. Dündar, “A comprehensive analysis on differential cross-coupled CMOS LC oscillators via multiobjective optimization,” in Integration, the VLSI, vol. 67, pp. 162-169, July 2019.

[26]. F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386-408, 1958.

[27]. D. E. Rumelhart, G. E. Hinton and , R. J. Williams, "Learning representations by back-propagating errors." *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.

[28]. M. Ding and R. I. Vemur, “An active learning scheme using support vector machines for analog circuit feasibility classification,” 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design, Kolkata, India, 2005, pp. 528-534.

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

[30]. T. O. Çakıcı, G. İslamoğlu, Ş. N. Güzelhan, E. Afacan and G. Dündar, “Improving POF Quality in Multi Objective Optimization of Analog ICs via Deep Learning,” 2020 European Conference on Circuit Theory and Design (ECCTD), Sofia, Bulgaria, 2020, pp. 1-4.

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

[32]. H. Liu and et. al., “Remembrance of circuits past: macromodeling by data mining in large analog design spaces,” in *Proceedings of the 39th annual Design Automation Conference*, 2002, pp. 437–442.

[33]. G. İslamoğlu, T. O. Çakici, E. Afacan and G. Dündar, “Artificial Neural Network Assisted Analog IC Sizing Tool,” 2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Lausanne, Switzerland, 2019, pp. 9-12.

[34]. K. Hakhamaneshi, N. Werblun, P. Abbeel and V. Stojanović, "BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with Deep Neural Networks," *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Westminster, CO, USA, 2019, pp. 1-8.

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

[36]. N. Lourenço and et. al., “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, 2019, pp. 13–16.

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

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

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

[40]. H. Wang and et. al., “Learning to design circuits,” arXiv preprint arXiv:1812.02734, 2018.

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

[42]. K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi and B. Nikolic, "AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2020, pp. 490-495.

[43]. M. Grabmann, F. Feldhoff and G. Gläser, “Power to the Model: Generating Energy-Aware Mixed-Signal Models using Machine Learning,” 2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Lausanne, Switzerland, 2019, pp. 5-8.

[44]. Kingma, D. P. & Ba, J. (2014), 'Adam: A Method for Stochastic Optimization' , cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[45]. https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5.

[46]. Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

[47]. Glorot, Xavier & Bengio, Y.. (2010). Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research - Proceedings Track. 9. 249-256.

[48]. F. Ertam and G. Aydın, "Data classification with deep learning using Tensorflow," 2017 International Conference on Computer Science and Engineering (UBMK), 2017, pp. 755-758.

[49]. H. Lee and J. Song, “Introduction to convolutional neural network using Keras; an understanding from a statistician,” Communications for Statistical Applications and Methods, vol. 26, no. 6, pp. 591–610, Nov. 2019.

[50]. Greff, Klaus & Srivastava, Rupesh & Koutník, Jan & Steunebrink, Bas & Schmidhuber, Jürgen. (2015). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems. 28. 10.1109/TNNLS.2016.2582924.

[51]. R. Martins, N. Lourenço, N. Horta, S. Zhong, J. Yin, P.-I. Mak, and R. P. Martins, “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,” in IEEE Transactions on Circuits and Systems I: Regular Papers (IEEE TCAS-I), vol. 67, no. 11, pp. 3965-3977, Nov. 2020.