Deep Reinforcement Learning applied to Analog Integrated Circuit Sizing

Tomás Bessa de Curado Rodrigues

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Nuno Cavaco Gomes Horta

Co-supervisor(s): Doctor Nuno Calado Correia Lourenço

Examination Committee

Chairperson: Prof. Pedro Filipe Zeferino Aidos Tomás

Supervisor: Prof. Nuno Cavaco Gomes Horta

Member of the Committee: Doctor António Manuel Lourenço Canelas

November 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

This work was hosted at Instituto de Telecomunicações, funded by Fundação para a Ciência e Tecnologia–Ministério da Ciência, Tecnologia e Ensino Superior (FCT/MCTES) through national funds and, when applicable co-funded European Union (EU) funds under the project UIDB/50008/2020.

The most heartfelt thank you to my co-supervisor, Professor Nuno Lourenço, who gave me the encouragement and guidance to complete this work. Without his help, this thesis could not have been written. Professor Nuno Lourenço was always available to clarify any question I had and assist me with any problem I faced throughout the entire development of this work.

Thank you to my supervisor, Professor Nuno Horta, for guiding me in this work, his assistance was crucial to the development of this thesis.

I would like to thank my little brother, Duarte Bessa, the most important person in my life, for always being there for me, through every obstacle I faced and every joy that came with this work. He is, without a doubt, the best brother I could have asked for.

To my parents Maria Curado and Carlos Bessa, who were very supportive throughout the entire process, giving me strength and love. I would also like to thank my grandfather, António Curado (Vôvas) and my grandmother Isabel Bessa for their love and support throughout this path.

Thank you to my girlfriend, Ana Rodrigues, for being there for me when I needed a push to get back up, for hearing all my thoughts and provide me with heartfelt support, no matter what.

To my best friends, Jaime Marques, and Bárbara Andrade, the most grateful thank you for their support. They give me inspiration every day to become a better version of myself. Without them, I could not have done this thesis, so thank you so much.

I would like to thank my friends at Instituto Superior Técnico, Rodrigo Vieira, Ricardo Santos, Inês Ferreira, and André Amaral. They helped me take my mind off the problems and focus on the progress instead of the obstacles. They helped me when I had any questions and offered their support constantly.

I would like to give a special thank you to my friends Pedro Gil, António Godinho, Marta Martins, Inês Bolhaqueiro, Joana Carolina Matias, Mariana Ribeiro, Olavo Freitas, and Beatriz Rosalino for their patience, support, and love.

I would like to give a special thank you to my “family” in Lisbon, the most amazing friends anyone could ask for: Pedro Furtado, Sofia Guerreiro, Diana Costa, João Santos (Parreco), João Barbosa, Pedro Fialho, and Pedro Silva.

Finally, the most heartfelt thank you to *Fairy Tail* (anime show) and its soundtrack. *Fairy Tail* had so much influence over the course of this work that it needed to be mentioned here. Not only did *Fairy Tail* helped me maintain my mental health with its music, it gave me passion and commitment to conquer every obstacle that came my way, just like it taught me. *Fairy Tail* was truly indispensable and I am glad it helped me so much.

Resumo

Este trabalho de tese de mestrado está inserido na área de Automatização de Projetos Eletrónicos. Esta é uma vasta área que cobre diversos temas de extrema importância para as atuais e futuras gerações da sociedade. Neste trabalho foca-se na área de dimensionamento automático de circuitos integrados e como o processo pode ser otimizado e melhorado. Com esse objetivo, uma nova abordagem é introduzida. Esta abordagem inovadora tem como base técnicas de Aprendizagem Profunda, Redes Neuronais e Aprendizagem Reforçada.

Atualmente, a ausência de soluções globais padrão para a problemática de dimensionamento automático de circuitos analógicos leva a um fastidioso, complexo e demorado processo de design, contrastando drasticamente com os circuitos digitais. Recentemente, tem havido diversos desenvolvimentos por forma a combater as disparidades ao nível de performance entre os circuitos digitais e analógicos. Este tópico tem sido objeto de discussão intensiva, tendo havido progressos notáveis, o que é suportado pela publicação de diversos artigos pela comunidade científica.

O principal foco deste trabalho consiste em aplicar Aprendizagem Automática, assim como técnicas de Aprendizagem Profunda, ao dimensionamento de circuitos integrados analógicos. Em particular, é destacado o algoritmo Ator-Crítico de Vantagem, um método que pode ser aplicado ao dimensionamento de circuitos integrados analógicos. Recorrendo a ferramentas do software AIDA, o agente proposto tem como objetivo melhorar o funcionamento do AIDA usando uma solução alternativa de aprendizagem reforçada para o dimensionamento de circuitos integrados analógicos. O modelo foi aplicado a duas topologias diferentes de circuitos, sendo elas um VCOTA e um FCA. Para ambas as topologias o agente foi capaz de dimensionar com sucesso os componentes rapidamente e eficientemente de modo a cumprir uma série de especificações definidas previamente que condicionam a sua performance.

Palavras-chave

Automatização de Projetos Eletrónicos, Circuitos Integrados Analógicos, Redes Neuronais Artificiais, Aprendizagem Reforçada, Aprendizagem Profunda, Ator-Crítico de Vantagem.

Abstract

The following work resides in the scientific field of electronic design automation. This is a vast area which covers several different subjects of extreme importance to current and future society. There is a special emphasis on the automatic sizing of integrated circuits and how the process may be improved and optimized. Having this goal in mind, a new approach will be introduced. This innovative approach makes use of Deep Learning techniques, Artificial Neural Networks and Reinforcement Learning.

Currently, the absence of standard and widespread solutions for automatic analog integrated sizing leads to a cumbersome and complex design, not to mention time-consuming, severely contrasting with their digital counterparts. To fight the productivity gap between analog and digital IC design, there have been, recently, several developments in this area. It is being subject of intensive research considerable progress has been made, which is supported by the publishing of various scientific articles.

This work mainly focuses on a relatively recent idea, which is applying Machine as well as Deep Learning techniques to analog IC sizing, namely Reinforcement Learning methods, as it is stated earlier. It presents Asynchronous Actor-Critic, a RL approach which can be applied to analog IC sizing. By resorting to AIDA software tools, the proposed algorithm aims to enhance AIDA's flow using an alternative RL-based sizing solution. The model is applied to two different circuit topologies, VCOTA and Folded Cascode. For both cases, the agent is capable of successfully performing the sizing of their components in a short time while fulfilling set target specifications.

Keywords

Advantage Actor-Critic, Analog Integrated Circuits, Artificial Neural Networks, Deep Learning, Electronic Design Automation, Reinforcement Learning

Table of Contents

List of Figures

List of Tables

Acronyms

- AAC - Asynchronous Actor-Critic
- ACD - Analog Circuit Design
- AI - Artificial Intelligence
- AMS - Analog Mixed-Signal
- BO - Bayesian Optimization
- CAD - Computer Automated Design
- CAN - Circuit Attention Network
- DDPG - Deep Deterministic Policy Gradient
- DNN - Deep Neural Network
- EDA - Electric Design Automation
- FoM - Figure of Metric
- GCN - Graph Convolutional Network
- GNN - Graph Neural Network
- IC - Integrated Circuit
- LNA - Low-Noise Amplifier
- MCTS - Monte Carlo Tree Search
- ML - Machine Learning
- MLP - Multi-Layer Perceptron
- NF - Noise Figure
- MS - Mixed Signals
- PGNN - Physics-guided Neural Network
- PPO - Proximal Policy Optimization
- ReLU - Rectified Linear Activation Unit
- SoC - System on Chip
- TRM - Trust Region Method
- TRPO - Trust Region Policy Optimization
- UCB - Upper Confidence Bound

1 Introduction

The first chapter is dedicated to the introduction of Analog Integrated Circuits (IC). Analog Integrated Circuit properties will be highlighted, and there is an emphasis on the challenges regarding optimization-based methods, which concern sizing approaches. To address the problem, there is an introduction to the concept of Machine Learning (ML) and the possible use of Artificial Intelligence (AI) to automate Analog Circuit Design (ACD).

1.1 Motivation: The Analog IC Design Effort

Currently, markets are frenetic, a reality that has never been experienced before. Industries are developing and adapting to new patterns, and the market has become increasingly demanding. The electronics market, such as analog and digital circuits, is no exception. Developers, in general, often struggle with pursuing the challenge of creating better circuits with greater power and progressively smaller components, exploring the tradeoff between size and performance.

The overall projection of total IC sales growth in 2022 is unchanged and expected to rise 11% this year to a record-high $567.1 billion. The new 2Q22 Update keeps the 2022 growth forecast unchanged in analog ICs (up 12%) and logic integrated circuits (up 11%)[1]. The growth associated with the Integrated Circuits industry will contribute to advances in several different areas, namely education, healthcare, and transportation. With the evolution of worldwide communications, like the Internet, which combines several electronic components, developing such systems has become crucial to keep up with society's needs.

More often than not, analog IC design is being done manually, aiming to meet several specifications simultaneously, resulting in long design cycles and challenging tasks. Therefore, experts are progressively adopting digital components as much as possible.

Digital components are usually preferred over their analog counterparts for various reasons. First, digital circuits are better supported and easier to reuse, and many automated tools assist in designing digital circuits [2]. Therefore, it is generally preferable to use digital circuiting, and developers are progressively switching due to the before mentioned advantages.

However, this type of circuit cannot be abandoned since real-world interactions are made through analog and radio frequency (RF) circuits. Hence, there is a need to combine both types of circuits. The result of this combination is named Mixed-Signals (MS) Systems on Chip (SoC). Analog circuits still constitute a small fraction of MS-SoC, and they are the ones that require the most effort to be built, even though they end up occupying less area in MS-SoCs, as portrayed in Figure 1.1.

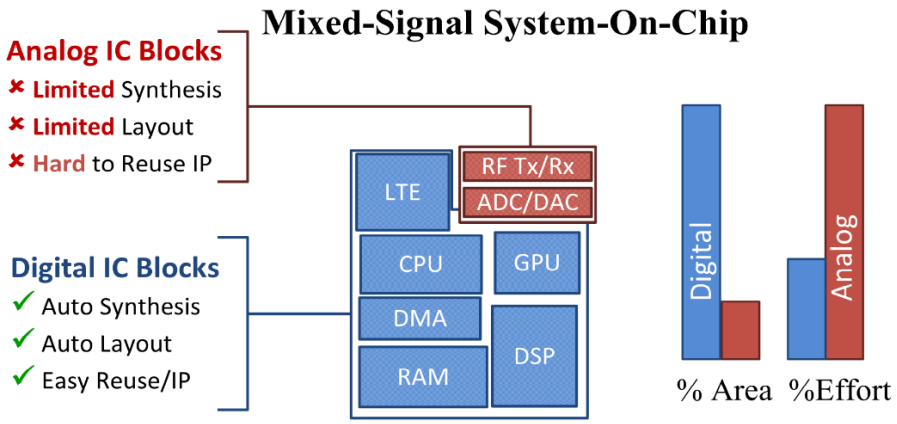

Figure 1.1 - Comparison between Analog and Digital Integrated Circuits [2]

As visible from the image in Figure 1.1, MS-SoC are mostly constituted by digital components due to reasons stated earlier. Analog IC composes roughly 20% of the total area, even though the effort required to produce them is significantly larger than digital ICs. Analog circuitry deals with continuous signals and is more susceptible to various types of noise, parasitic disturbances, crosstalk, etc. The leading cause behind this large difference is associated with analog intrinsic properties. Namely, it is generally less systematic, more intricate, and knowledge-intensive than digital ICs. These factors are critical when digital and analog circuits are integrated into the same design.

Electronic Design Automation (EDA) tools and design methodologies have been improved to cope with the latest IC technologies. However, there is still some road to cover, more specifically, the fact that they are still not balanced (Analog IC and Digital IC), the latter having more resources available at its disposal.

Such a gap is probably due to the difference in market size. Since digital circuitry has much more demand, the market around it grows more prominent than its analog IC counterparts. Still, analog ICs are responsible for expensive re-runs and design errors, which leads to investment in tools to mitigate the design effort while ensuring reliability.

To summarize, two main reasons justify the considerable development time of analog circuits when compared to digital circuits [3]:

- At the moment, analog IC blocks are being integrated using technologies that are destined for the optimization of digital IC's;
- Analog circuit blocks are complicated to reuse due to the high sensitivity regarding the environment, nearby circuits, and overall process variations;

1.2 Topic Overview: Analog & MS Design Flow

The specific design flow of Integrated Circuits is often unique and specific to each designer or company producing the IC, like the concept of human fingerprints. Each designer/company has its style of design. However, the more significant part of the work related to analog design flow can be generally mapped into the model proposed by Gielen and Rutenba [4]. This model is described in Figure 1.2. It demonstrates the design flow for analog mixed-signal IC as a series of top-down design steps

repeatedly executed from the system level to the device level and bottom-up layout generation and verification.

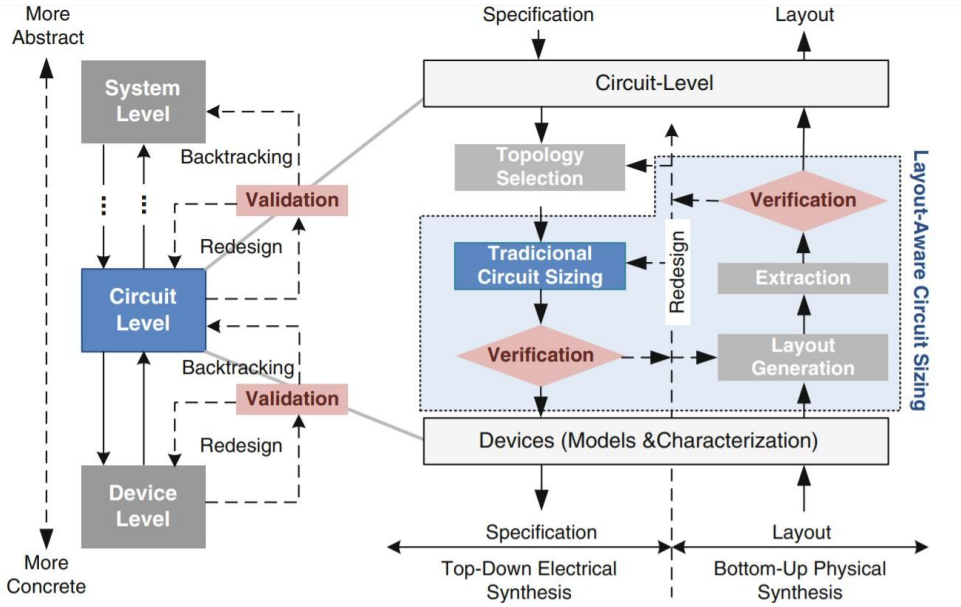

Figure 1.2 - Hierarchical level and design tasks of design flow architectures [2]

Adopting the top-down strategy proved advantageous since it made it possible to explore system architectures of higher complexity, which led to improved system optimization at a higher level of abstraction. This can be achieved before the beginning of more specific and intricate implementations at lower levels. This way, finding problems at the beginning of the design process is possible, leading to an increase in first-time success rate and a decrease in the necessary time it takes to conclude the entire process.

The system complexity influences the number of hierarchy levels in the design flow. A generally accepted representation of design architecture consists of two prominent design paths:

- **Top-down electric synthesis**: Includes topology selection, design verification, and specification translation. The latter is also known as circuit sizing at the lowest level;
- **Bottom-up physical synthesis:** Includes detailed design verification after layout extraction as well as layout generation;

The process of determining the circuit topology best suited for a given problem is called topology selection. When selecting the topology that better suits a given problem, it is crucial to meet the specifications at the current hierarchy level, where the available topology can either be chosen from a determined set or synthesized. Each block carries its specifications to the next level down the hierarchy, and the process is repeated until the top-down electric synthesis flow is complete.

The task which ensures the mapping of high-level block specifications into independent specifications to each sub-block is named Specification Translation. This task can be narrowed down to circuit sizing when dealing with the lowest level, as the sub-blocks are single devices. Circuit sizing is an iterative process that determines a suitable set of lengths, widths, and multiplicities for each device in the topology to achieve desired specifications.

1.3 Automatic Analog IC Sizing

Recent developments show that analog and RF cells, such as amplifiers or oscillators, can be automatically synthesized from specification to fabrication. The most common methodologies follow an optimization-based strategy with accurate circuit simulations in the loop to evaluate the circuit's performance. These tend to be time-consuming since the complex relationship between design parameters and circuit specifications plays a big part in complicating this task but can find usable solutions without user intervention.

AIDA-C[3], developed at Instituto de Telecomunicações (IT), is one such optimization-based sizing approach. It is part of the AIDA Framework, an electronic design automation framework fully developed at IT, and appears as an Electronic Design Automation tool to aid designers in doing their job better and faster. AIDA multi-objective design methodology for automatic, analog IC sizing is based on NSGAII multi-objective multi-constraint optimization, and the circuit's performance evaluation is done with circuit simulators, e.g., ELDO, Spectre, or NGSPICE, ensuring that the developed automatic circuit sizing is compliant with the accuracy requirements of the analog designers.

In AIDA-C, circuit sizing and optimization are implemented as a multi-objective multi-constraint optimization problem defined as:

$$\begin{aligned} &\textit{find } x \textit{ that minimize } f_m(x) && m = 1,2,...M \\ &\textit{subject to } g_j(x) \geq 0 && j = 1,2,...J \\ &x_i^L \leq x_i \leq x_i^U && i = 1,2,...N \end{aligned} \tag{1.1}$$

where, x is a vector of N optimization variables, $g_j(x)$ one of the J constraints on the circuit performances and $f_m(x)$ is one of the M circuit performances being optimized. However, these optimization-based sizing approaches require many circuit simulations, which take time and consume energy. Hence the search for new and more effective approaches to automatically size analog ICs. Here is where Machine Learning advances come in handy, presenting significant innovation potential.

1.4 Goals

This work builds upon AIDA and explores using Reinforcement Learning (RL) techniques for automatic Analog IC sizing. Reinforcement Learning emerged as a possible approach to analog IC in recent years.

Therefore, the main goals established for this work are the following:

- Use the same circuit setup as AIDA
- Encapsulate AIDA circuit setups in environments suitable for Deep RL
- Develop a Deep RL approach to analog IC sizing using Asynchronous Actor-Critic (A2C)
- Experiment in Analog Circuits, which will be later applied and tested on VCOTA and Folded Cascode.
- Create an environment based on an interface with the OpenAI Gym™ simulator. Not only will this contribute to this works development but also future works on different agents as well.

- Finally, this work will resort to AIDA [3], an analog IC design automation environment developed at IT, as an alternative to the utilized evolutionary algorithm. If successful, the objective is to generalize the algorithm to be applicable to other circuits.

1.5 Document Structure

The current document is organized as follows:

- Chapter 2: This chapter is dedicated to the presentation of the state-of-the-art. The goal of this chapter is to provide an analysis of the work that exists nowadays regarding this matter. It also covers the approaches that are currently available worldwide and there is a detailed description of their advantages as well as their drawbacks. Additionally, a comparative analysis is presented, where the discussed approaches are compared;
- Chapter 3: In the third chapter there is a thorough description of the proposed Deep RL sizing approach. The goal of this sections is to provide an insight on the agent's characteristics and how it can integrate AIDA's flow by replacing the optimizer. Additionally, the sizing process is explained in detail to provide a clear idea on how the entire process works;
- Chapter 4: The fourth chapter is where the results are presented. In this section, the application of the work developed and described in Chapter 3 is tested and the results are evaluated. This way, it is possible for the reader to visualize how the proposed approach handled the analog IC sizing as well as how the algorithm reacts to different circuit parameterizations and target constraints;
- Chapter 5: The final chapter is destined to the presentation of the conclusions derived from this work. A final appreciation is given and a summary of the impact of this approach is provided. Additionally, future work is discussed;

2 Background and Related Work

The problem inherent to Analog IC design is a complex one. It is time-consuming and knowledge-intensive, and manual design is becoming unbearably complex. Design automation solutions would help to manage the design complexity, making the design less time-consuming and practical, but for analog ICs, automatic design tools are not widely used, and no standard automation flow exists. Therefore, several approaches emerged in the scientific community to provide a timely, trustworthy solution.

Regarding analog IC sizing, ML, particularly RL, starts to be explored, and this chapter covers all relevant recently reported results to support the development of the work presented here. Firstly, an overview of the concept of how analog circuit sizing approaches work is given, followed by a special focus on AIDA. Afterward, an overview of Deep RL is provided, which is followed by a description of how Deep RL is being used on analog IC sizing in scientific work. Finally, a comparative analysis of the evaluated works is done, and the developed RL approach is defined.

2.1 Analog Circuit sizing approaches

The design of analog IC is time-consuming due to the non-linear relationship between the design parameters and circuit/device specifications. Generally, a handmade calculation can restrain the design space and provide a good starting point for the designer. However, the process is cumbersome due to the many iterations it takes to achieve the expected specifications. Regarding automation, the solutions proposed over time follow two main trends: *knowledge-based sizing* and *optimization-based sizing* [10]. Knowledge-based sizing relies on expert knowledge, using tools like IDAC [11] and BLADES [15]. With the assistance of design equations and a design strategy, tools like these can reproduce a pre-designed plan. This approach shows, overall, favorable results concerning automatic analog IC sizing. The main advantage is the short execution time. However, deriving the design plan is a complex task requiring much time. The constant supervision that needs to take place to keep the design plan up to date and according to technological breakthroughs is what makes this approach suitable only as a first-cut design.

Optimization-based sizing can be further categorized into model-based methods and simulation-based. Model-based approaches use simplified equations/polynomials, which are used to model circuit performances. Then, they are used to solve optimization problems based on the calculated performance

results of those models. Since this is a theoretical approach, the predicted performances deriving from circuit models are not real-circuit performances.

Model-based approaches cannot ensure accuracy in their results, especially when considering large-scale circuits. Simulation-based methods significantly differ from their counterparts. They rely directly on simulation results and are, therefore, more accurate in most cases. However, simulation-based calculation relies on intensive computation and is a time-consuming process, which increases in complexity even further when processing larger circuits, resulting in an enormous time cost during the optimization process (Table 2.1 - Comparative analysis between Model-based approaches and Simulation-based approaches).

Table 2.1 - Comparative analysis between Model-based approaches and Simulation-based approaches

Model-based	***Simulation-based***
Utilize equations/polynomials to model circuit performances	Relies on simulation results
Based on theoretical expressions which compromises its accuracy	Intensive computation makes it more reliable and accurate
Unable to ensure accurate results, especially in large-scale circuits	Very expensive and time-consuming process (Scales even further in large circuitry)

Several works have been proposed where optimization-based methodologies are endorsed as a strategy to design and achieve optimal analog automatically and RF circuits [29]-[44]. The majority only tackles simple circuits like LNAs and VCOs and cannot handle more complex designs [29]-[36].

AIDA-C, whose core optimization engine is illustrated in Figure 2.1, uses a modified NSGA-II state-of-the-art multi-objective multi-constraint optimization kernel, enabling the efficient exploration of design tradeoffs. At the same time, the inclusion of corner cases and the usage of circuit simulators (Eldo®[12][13][14]) ensures the accuracy and reliability of the solutions.

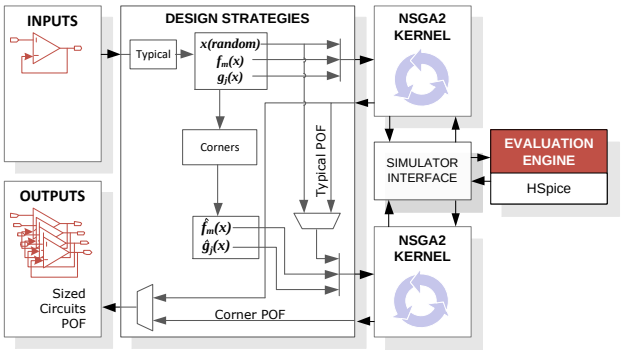

Figure 2.1 - AIDA-C Architecture [3].

The circuit design is then optimized by an NSGA-II multi-objective multi-constraint kernel [6], using simulated binary crossover and mutation operators [6], tournament selection, and constrained-based dominance check, modified to evaluate the fitness of the individuals using the circuit simulator. The design variables define the search space, and the state representation is defined by an array of real values (the current design).

AIDA minimizes the M objective functions $f_m(x)$ as defined in (2.1), where the measured circuit characteristic, pm, minimized (e.g., Current consumption, Area, etc.) are used directly, while the ones being maximized (e.g., DC gain, Unity Gain Frequency, etc.) are multiplied by −1. The constraints are handled in the form $g_i(x) \geq 0$, and are obtained from the designer specification, e.g., DC gain larger than 50 dB, Current consumption smaller than 80 µA, etc., according to (2.2), where pi is the measured circuit characteristic, and Pi is the constraint limit.

$$f_m(x) = \begin{cases} p_m & \text{when minimizing } p_m \\ -p_m & \text{when maximizing } p_m \end{cases} \tag{2.1}$$

$$g_i(x) = \begin{cases} \dfrac{p_i - P_i}{|P_i|} & \text{when the constraint is } p_i \geq P_i \\ p_i & \text{when the constraint is } p_i \geq P_i \text{ and } P_i = 0 \\ -p_i & \text{when the constraint is } p_i \leq P_i \text{ and } P_i = 0 \\ \dfrac{P_i - p_i}{|P_i|} & \text{when the constraint is } p_i \leq P_i \end{cases} \tag{2.2}$$

The constrained dominance criteria in AIDA's (the same as NGSA-II) sums all the constraints that are not met, *gsum*, and uses the value for the primary sorting criteria.

$$gsum(x) = \sum_{j=1}^{J} c_j\ (x) \text{ with } c_j\ (x) = \begin{cases} 0 \text{ if } g_j\ (x) \geq 0 \\ g_j\ (x) \text{ if } g_j\ (x) < 0 \end{cases} \tag{2.3}$$

While evolutionary optimization has shown to be capable of globally exploring the analog IC design space for a wide range of circuit designs, the execution time is still a significant concern due to the demanding circuit simulations. In this context, the use of ML to increase sampling efficiency is being widely explored.

For example, in [8], a classifier first predicts if the solution will likely be valuable to the optimization. Only those solutions that are promising are passed to the circuit simulator. Other approaches, such as [], use Bayesian optimization to reduce the number of needed simulations by making better guesses of the optimal solution in the design space. Alternatively, reinforcement learning, the focus of this work, is also being explored as a suitable technique for analog IC sizing.

2.2 Reinforcement Learning for Analog IC Sizing

Reinforcement Learning is a ML technique that uses a reward-based methodology, i.e., it rewards an agent based on its behavior. For instance, if an agent performs the desired action, it is rewarded. However, if the performed action/behavior is undesirable, it is punished. Generally, an RL agent can successfully perceive and interpret its surroundings, which form its environment. The agent can act and learn further based on what is previously learned.

The learning process is illustrated in Figure 2.2. When an agent is at a concrete state (s ∈ S) and proceeds to take a suitable action (a ∈ S), it obtains an immediate reward (r ∈ R) from the environment based on the results of said action. The main goal behind the RL loop is to maximize the reward function. In each problem, independent of what it entails, an agent is unaware of whether it will find a solution. If it does eventually find a valid solution, it will get instantly rewarded and will know, for future occasions,

that if it takes the path that previously led it to the solution, it will get a guaranteed reward once more. However, the found solution is not assured to be the optimal solution. Hence, repeatedly taking the same path is likely to prove unproductive.

Therefore, the agent must have a certain degree of freedom to explore different paths and eventually achieve a better solution, resulting in better rewards. During the discovery process, the agent can struggle to improve the current solution, resulting in expensive paths and excessive computation. Considering all these factors, finding a balance between exploring the environment in search of a new solution and ensuring the agent gets a reward is crucial.

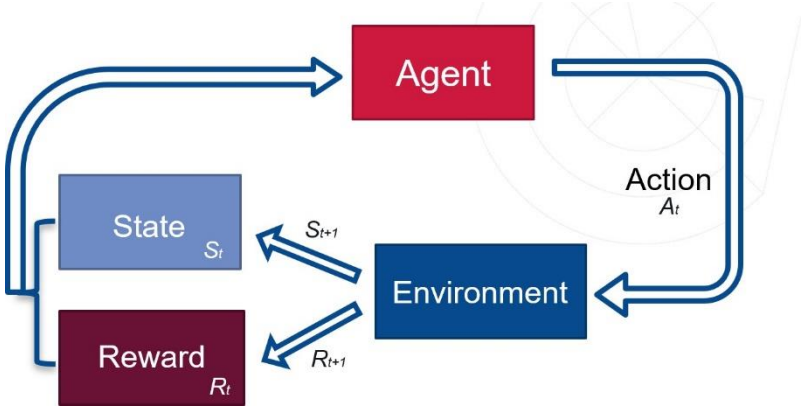

Figure 2.2 - Reinforcement Learning loop

There are several approaches involving Analog IC sizing, which incorporate Reinforcement Learning [22][23][25][47][48][50].The following subsections review several state-of-the-art works that use RL for analog IC sizing.

2.2.1 Deep Reinforcement Learning for Analog Circuit Sizing (2020)

In [10], a Deep RL simulation-based approach is taken with an additional symbolic filter to reduce the amount of circuit simulation. It relies on simulation results to assign rewards directing the learning process. In addition, a Deep Neural Network (DNN) is employed. As for the algorithm, it learns and improves over successive trials and respective rewards. Its goal is to find an optimal solution by discovering which parameter needs an alteration and by which amount. To mitigate the expensive runtime, a Symbolic Filter quickly evaluates the performance, filters trials, and eliminates those who do not meet target specifications.

Circuit parameters, such as biases, component sizes, resistances, etc., can be classified into fixed or changeable. The state is represented by an array, which contains every circuit parameter. The action space is defined as an increment/decrement in the value of a parameter in the state array. The amount associated with each change will depend on the product of two user-design hyperparameters, change rate, and change capacity.

If by any means, an action breaks any implemented constraint (for example, for the two-stage OpAmp, if V_{DD} is smaller than both V_{INP} and V_{BP}), it is automatically considered invalid. By only considering valid actions, unreasonable combinations of parameter values are eliminated, which translates into a reduction in the number of simulations and, consequently, an improvement in the

efficiency of the sizing process. This RL approach encodes design objectives into the rewards to instruct the learning direction.

The Symbolic Filter is the component that has had the more significant impact. Its goal is to apply symbolic analysis to evaluate partial performance attribute factors. Among the parameters encoded in the state are sizes and biases. Both serve as inputs to the Symbolic Filter. However, the symbolic analysis needs all parameters relative to all involved transistors, thus creating a gap. The proposed solution involves the curve-fitting technique. The curve-fitting technique is used to derive the transistor models that reflect the parameter mappings and obtain the missing parameters.

The results were obtained on a CMOS 65nm process from an Operational Amplifier (OpAmp) circuit, the folded cascode OpAmp. As for the experiment performed, each episode contains 50 steps, and, as a result, 50 tuples are produced. Each tuple includes states, actions, and rewards, and it is produced by the Physics-guided Neural Network (PGNN).

The reward function's weights assigned to performance attributes are: 0.3 for dc gain and phase margin, 0.2 for unit gain bandwidth and gain margin. Initially, set biases and sizes are not enough to put all transistors into the desired operating modes. However, there is a significant improvement, namely regarding solutions with good performance. As for the learning rate, Figure 2.3 demonstrates the process extremely well.

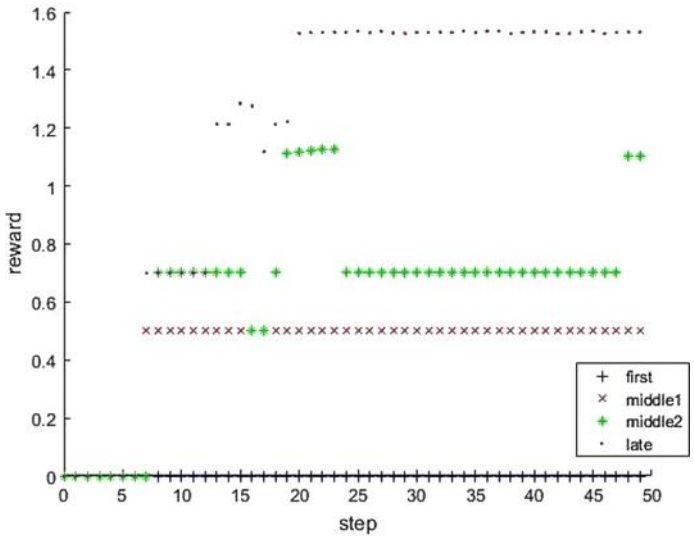

Figure 2.3 - Rewards obtained at different periods [5]

In the early period, every trial got a reward of 0, since the size values did not fulfill the constraints set by the symbolic filter. Then, in the first middle period, a few trials pass the symbolic filter, and few non-zero rewards emerge, proving that the agent gets experience from previous learning. As it progresses, reaching the second middle period, where rewards become larger and feasible solutions start to appear. Finally, in the late period, rewards grow larger, indicating that the performance is greatly improved.

2.2.2 A Circuit Attention Network-Based Actor-Critic Learning Approach to Robust Analog Transistor Sizing (2021)

In [18], a sizing approach built upon an Actor-Critic framework - DDPG (Deep Deterministic Policy Gradient). Its main goal is to address the robust circuit sizing problem. Actor-Critic methods are

temporal difference methods that have a separate memory structure to represent the policy value function. The policy structure may also be known as the actor since it is used to select actions. On the other hand, another model estimates the value (the expected reward of the state), also known as the critic, because of it" criticizing" the actions previously performed by the actor.

The approach is divided into three stages: Circuit Attention Network (CAN),Stochastic technique, and validation of results. The CAN serves as both the Actor and Critic and is, at heart, a customized graph NN. It allows knowledge transfer between different topologies of the same circuit type, and its effectiveness has been proved through simulations. The stochastic technique is implemented to mitigate the layout effect. It tries to turn a sizing solution robust against layout uncertainty but requires a previously sized circuit. Result validation is done through post-layout simulation, showing significant improvement in circuit performance compared to Bayesian Optimization and GCN-RL. Figure 2.4 shows a diagram which illustrates the proposed approach:

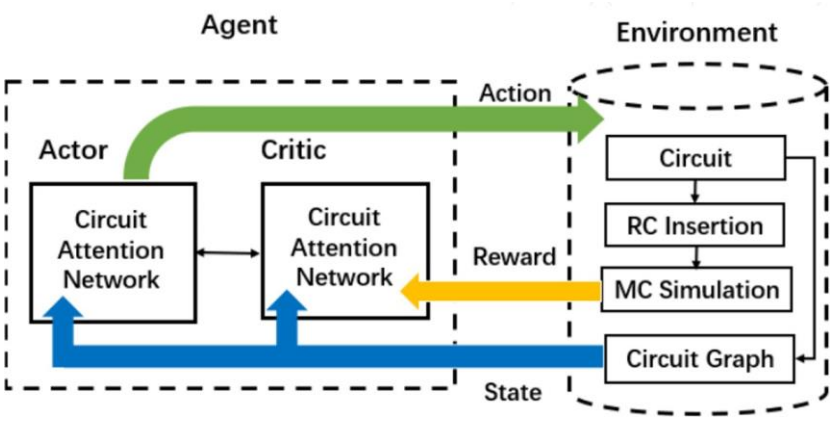

Figure 2.4 - Robust transistor sizing flow [18]

The robust transistor sizing problem is solved using DDPG, as it is one of the most advanced RL methods. In this approach, the state space is defined by transistor sizes x in their legal range $[x_L, x_U]$. On the other hand, the action space is defined by transistor size changes according to $\mathbf{a} = [a_1, a_2, ..., a_N] \in \{-1, 0, +1\}^N$. This way, the size of each transistor may be incremented, decremented, or maintain its value (multiple transistors can change size simultaneously in one step). The reward function is a modified Figure of Merit (FOM_R) defined in (2.4), where μ stands for mean and σ for standard deviation, β being the weighting factor.

$$FOM_R = \sum_{j=1}^{M} w_j(\mu(q_j) - \beta \cdot \sigma(q_j)) \quad (2.4)$$

In short, the process is the following: An action is performed based on information provided by the actor. The circuit is then modified accordingly. Then, the kernel (deep NN as the underlying structure of an analog circuit) of the stochastic technique addresses the layout effects. Next, RC circuits and Monte Carlo simulations compute the rewards FOM_R. Finally, the value function $Q(x, a)$ is updated for the critic function.

A tradeoff between robustness and layout uncertainty is considered when computing the rewards. It is the kernel of the stochastic technique for considering layout effects during sizing. Its main advantage is preventing expensive processes such as layout, extraction, and post-simulation. So, RC

(resistance and capacitance) elements are inserted for any given sizing solution to emulate the layout effects. After that, they are simulated using Monte Carlo, and the weights are obtained to calculate the FOM_R.

It was earlier stated that the CAN, a customized Graph Neural Network (GNN), serves as both actor and critic. First, it is important to summarize the circuit graph and features for CAN. It basically transforms an input constituted of circuit features and devices into a graph, to be analyzed by the network. Devices serve as nodes and edges indicate connections between nodes. For a better visualization, there is a representation of the process in Figure 2.5.

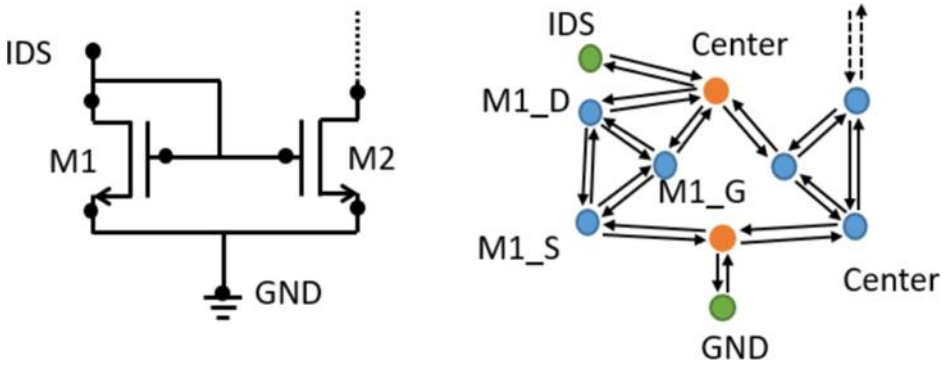

Figure 2.5 - Current mirror and its graph [18]

As for its architecture, the CAN is composed of a series of embedded layers followed by a Multilayer Perceptron (MLP). CAN adopts attention-based convolution in feature embedding and simultaneously performs graph pooling to capture a global view. Each feature embedding layer in the Circuit Attention Network is elaborated as follows:

1. Attention construction and compression
2. Graph Convolution
3. Node and edge pooling

A critical aspect of the CAN is its ability to transfer knowledge, i.e., it is applicable to several different topologies. Usually, there are two problems associated with knowledge transfer. The first one is letting a single GNN accommodate different input dimension sizes resulting from different graph structures. CAN successfully solves the first problem since graph pooling in CAN performs node/edge feature dimension reduction and effortlessly handles the input of different dimensions. The other problem is making a single GNN model provide different-sized outputs due to the difference in action spaces. This setback is solved by the actor network of target topology. It inherits the feature embedding layers from the source topology, replacing the MLP of source topology with an untrained MLP, thus needing very few data samples.

Comparing the quality of the solution presented (the CAN-RL method with FOM_R reward) with the GCN-RL method (which uses regular FOM reward). The following table provides a head-to-head comparison between the methods for each feature. The" X" mark signals which method performs better:

Table 2.2 - Comparative analysis between the CAN-RL method and the GCN-RL method[18]

	CAN-RL	GCN-RL
Multiple-Layout design	X	
FOM values for multiple layout design	X	
Default Layout setting	X	
Average FOM values for default setting	X	
Schematic designs		X

As it is visible from the table, the CAN outperforms its counterpart in almost every aspect. All in all, CAN-RL is better in regard to circuit performance and robustness. In terms of convergence and knowledge transfer, the proposed method is tested differently. The convergence is compared between the former two methods (CAN-RL and GCN-RL) and the additional Bayesian Optimization (BO). Each method is composed of two variants: the first one uses layout- oblivious FOM as its reward and the second one utilizes layout-aware FOMR as its reward. The results are presented in Figure 2.6.

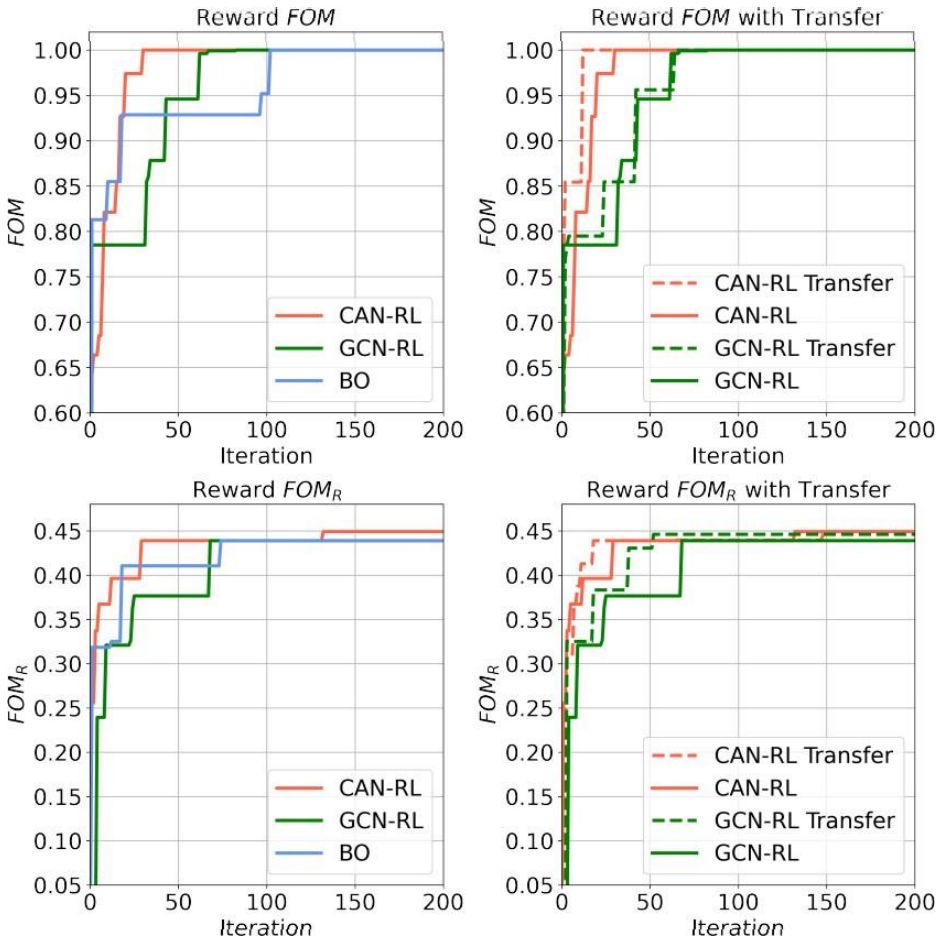

Figure 2.6 - Learning curves [18]

As visible from the figure, the red curve proves that CAN-RL converges faster than the both GCN-RL and BO in most cases. The dashed red curves, on the other hand, display further speedup for CAN-RL knowledge transfer. Regarding iterations, the convergence rate is similar when comparing FOM and FOM^R rewards.

Even though every method eventually converges and finds a similar solution, CAN-RL is the fastest one to converge. It converges faster than both GCN-RL and BO. Relatively to BO, CAN-RL converges 5.6 times faster for FOM rewards and 6.8 times faster for FOM^R rewards. When the reward function is FOM^R, the transfer makes CAN-RL approximately 2.8 times faster, which proves knowledge transfer facilitates further speedup.

2.2.3 AutoCkt: deep reinforcement learning of analog circuit designs (2020)

In [19], AutoCkt, emerges as a response to the necessity of finding a sample efficient, accurate, generalizable, and intuitive method without the overhead of constraint generation. It aims to find post-layout circuit parameters for any given target specification and gain knowledge of the entire design space using the sparse sub-sampling technique.

AutoCkt is a ML optimization framework that is trained by recurring to RL. The proposed method offers several different advantages. For example, it understands the design space in the same style as a circuit designer. In addition, it converges approximately 40 times faster than a regular evolutionary algorithm, resulting in less time-consuming processes. The algorithm for thismethod is shown in Figure 2.7:

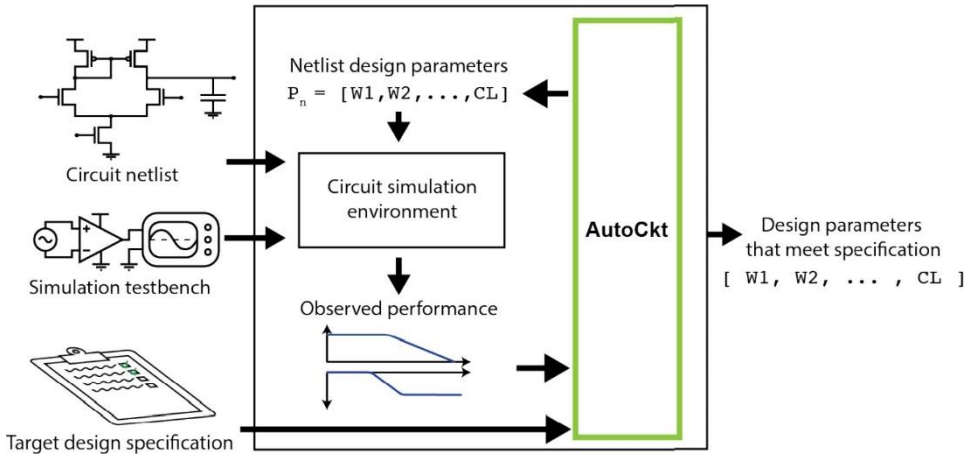

Figure 2.7 – Top level overview, showing what information is needed for AutoCkt in order to design any circuit topology to meet a given target design specification [19]

As far as the RL Agent is concerned, there are *N* parameters to tune to optimize *M* target specifications. The parameter space can be defined as $\mathbf{x} \in \mathrm{Z}^N$ and the design specification space as $\mathbf{y} \in \mathrm{R}^M$ (**y** is normalized to a fixed range). Concerning the generation of trajectories using AutoCkt, it processes as follows: The parameters are initialized to a given center point. Afterward, the NN uses the observed performance, target specifications, and current parameters to act. The action space is defined as an increment, a decrement, or a retaining in value. Training and deployment begin as the RL agent faces 50 randomly sampled target specifications. Afterwards, L trajectories are generated, and the targets are chosen among the target specifications. The rewards are cumulative as they depend on previous actions. Aiming to demonstrate AutoCkt's capabilities, three different simulation environments as well as three circuit topologies are considered. The first test is performed on a transimpedance amplifier. Multiple design objectives are set, and thetraining begins. After training is complete, the mean episode reward is greater than 0, which means the agent successfully learns how to reach positive goal states

over several target objectives. The second test consists of a two-stage operational amplifier, a more complex yet more common circuit. The total action space for this problem is approximately 10^{14} possible values, making random generation of parameters rather inconceivable. The set trajectory length assigned for the agent to converge is 30 simulation steps. The obtained mean reward reaches a value greater than zero as well and it does so in 25 ms, which makes the overall time tractable. The trained agent is run on 1000 randomly generatedtarget specifications.

The results are clear, AutoCkt can reach 963 out of 1000 target specifications and 40 times faster than a traditional genetic algorithm. The table below presents the results for the second test and compares how this method performed relative to a random RL agent and a traditional genetic algorithm.

Table 2.3 – Sample efficiency (SE) and generalization comparison table [19]

Metric	Opamp SE	TIA SE	Generalization Opamp
Genetic Alg.	1063	376	N/A
Random RL Agent	N/A	N/A	38/1000
[11]	27	15	963/1000

The third test involves a two-stage OTA where a negative g_m load is introduced, which is very similar to the second test. The circuit is more complex when it comes to its design and is more sensitive to layout parasitic. The action space for this test set has an order of complexity of approximately 10^{11} different parameter combinations. The results are like those of the previous two tests. It converges about40 times faster than a standard genetic algorithm, taking just 10 simulations to find a solution.

2.2.4 DNN-Opt: An RL Inspired Optimization for Analog Circuit Sizing using Deep Neural Networks (2021)

DNN-opt is one of the latest proposed RL-based approaches to analog circuit sizing. In [20], DNN-Opt emerged as an attempt to optimize previously known methods. DNN- Opt consists of a two-stage deep learning black-box optimization scheme, a fusion between RL's strengths, Bayesian Optimization (BO), and population-based techniques. Its key features include a two-stage Deep Neural Network (DNN) architecture inspired by actor-critic algorithms, a population-based search space control mechanism, and sensitivity analysis. This work can size large circuits efficiently. The overall framework of DNN-opt is presented in the figure below. The flow begins by generating samples in the design space. Afterward, a critic network is used to predict any new design point performance. The actor network uses the previously mentioned prediction and proposes new possible candidates for the simulation, a scheme like BO in space exploration.

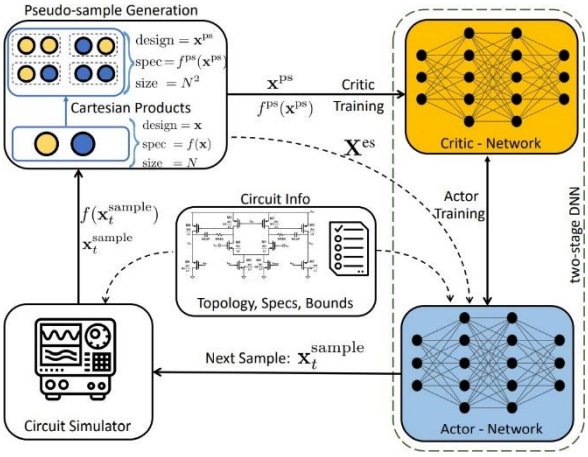

Figure 2.8 – DNN-Opt framework [20]

The two-stage network architecture uses a structure that does not differ from Deep Deterministic Policy Gradient (DDPG). However, to apply to an RL problem, it needs to be subjected to some alterations.

Some parameters are important to highlight to have a better understanding of the problem and its specifications. For starters, the state space involves mapping optimization parameters, which are circuit design variables. A state of k^{th} design is transformed as $s_k = x_k$. As for the action space, each action a_k is defined as a change in the optimization parameter vector x_k($a_k = \Delta x_k$). Before progressing to the functioning of the overall framework and the experimental results, it is important to describe the actor-critic network and how each of them is trained.

The proposed critic-network takes the circuit design variables as input (x, Δx), outputting performance predictions $Q(x, \Delta x|\theta^Q) \in R^{m+1}$ (one dimension is destined to objective specifications whereas m are for constraint specifications). In the problem context, the critic network is utilized as a way of modelling design variables to circuit performance relationships. To achieve effective training, data augmentation techniques are employed. By doing so, pseudo-samples are generated. Consequently, the input dimensions of the critic network change from d to $2d$. Further experiments have shown that training using pseudo-samples and the usage of $2d$ inputs boosts the network's accuracy significantly over acritic-network which uses d inputs and original samples. For N_b pseudo-samples, the used function to train the critic network is the Mean Squared Error (MSE).

Once the critic-network training is complete, it is possible to train the actor-network. Its training consists of searching the design space for *better* designs. The proposed approach aims to provide a change in the design parameter vector, translating in the expression $\Delta x_k = a_k = \mu(x_k|\theta^\mu)$. With the intent of objectively assessing the design quality, 58[20] proposed a Figure of Merit (FoM) function, $g(\cdot)$. The actor-network is trained using the function $g(\cdot)$ and replacing the SPICE simulation values by critic-network predictions $Q(x, \Delta x)$. Further, the used population is constituted of elite solutions to restrict the search space for the actor-network.

Summarizing, the overall framework for DNN-Opt is processed as follows: Firstly, sensitivity analysisis applied, causing a reduction in the number of design variables. Then, a random sample of points is collected to build the initial population. It is then followed by an initialization in actor-critic parameters and pseudo-sample generation. The next step is to train both the actor and critic networks,

followed by the construction of an elite population based on the FoM of total-population. Every elite population's design parameter serves as input to the actor-network and the output are proposed changes for them.

DNN-Opt was tested on two small building blocks: a folded cascode amplifier and a strong-arm latch comparator (implemented in 180*nm* CMOS technology). DNN-Opt is compared to Differential Evolution (DE), Bayesian Optimization with weighted expected improvement (BO-wEI) [8] and the GAS-PAD method [9]. DE has a simulation budget of 10000, whereas BO-wEI, GASPAD and DNN-Opt have a budget of 500 simulations. Statistics relative to each method are provided. For example, the *success rate* (number of times a feasible solution is found) and the evolution of the FoM are metrics used to compare the methods.

The first case is a folded cascode Operational Transconductance Amplifier (OTA). In this test, DNN-Opt shows high reliability, proving to find a feasible solution in all its trials. As opposed to DNN-Opt,BO-wEI and GASPAD performed relatively worse in comparison. Even though DE can find feasible solutions, it falls short to DNN-Opt since it does it 24 times more effectively. DNN-Opt design draws up to 43% less power on average. Its execution time is about 50 times smaller compared to the other methods. All of this results in more efficiency regarding runtime (about 2.5-16 times more). DNN- Opt also shows strong convergence behaviour (visible in the figure below), outperforming other methods. GASPAD can find the optimal FoM, even though it is slow, yet BO-wEI is trapped in local optima.

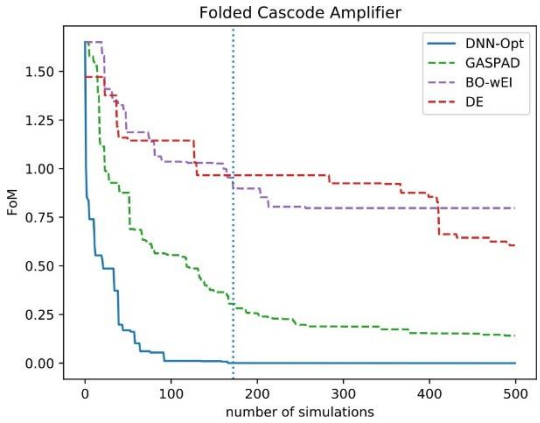

Figure 2.9 – The average FoM (lower is better) curve for 500 simulations[20]

The second case is relative to the Strong-Arm Latch Comparator. DNN-Opt is the only method that is able to find a feasible solution in every trial, showing 30 times more efficient when compared to DE. GASPAD does not fall behind DNN-Opt, still DNN-Opt finds a solution with 25% better power consumption. Regarding convergence in FoM, DNN-Opt finds a solution much earlier than other methods.

As far as large circuits are concerned, DNN-Opt still performs well, not being limited to small examples. The 4 large circuits tested were: Inverter Chain, Level Shifter, Low-Dropout (LDO) and Regulator, and Continuous-Time Linear Equalizer (CTLE). DNN-Opt is compared to a commercial black-box optimizer based on Simulated Annealing. DNN-Opt outperforms its counterpart in terms of the number of simulations required to meet the constraints,and the solution it presents is less costly than the one presented by the commercial optimizer. Table 2.4 demonstrates for both Simulated Annealing

(SA) and DNN-Opt the number of SPICE simulations that were performed for each model to meet the specified constraints. The data shows that DNN-Opt outperforms SA as it is faster at finding a solution when submitted to the same constraints as SA.

Table 2.4 – DNN-OPT results on industrial circuits [20]

Circuit	MOS	NODES	SA	DNN-Opt
Inverter Chain	8	7	>1000	**90**
Level Shifter	1.2k	3.9k	1200	**195**
LDO	167k	2.8k	552	**112**
CTLE	173k	63k	587	**150**

2.2.5 Trust-Region Method with DRL in Analog Design Space Exploration (2021)

One recent approach to analog IC sizing is presented in [21]. The proposed method consists of a general learning-based search framework. The contributions are distributed over three levels: system, algorithm, and verification.

On a system level, the proposition lies on a general framework for IC design space search. Fast migration is allowed by standardized API, providing well-formulated problems. Algorithm wise, RL is employed. It aims to directly replicate the dynamics of the SPICE simulation as opposed to the traditional process that is estimating cumulative rewards. Finally, verification is done by considering PVT conditions. Analog circuit sizing is usually formulated as a constrained multi-objective optimization problem.

The main goal of this problem in particular is to minimize the objective function c^{th} under the PVT condition *Minimize* $F_{m,c}(X)$. The X refers to the vector of variables which are set to be optimized. The general framework is depicted in the below:

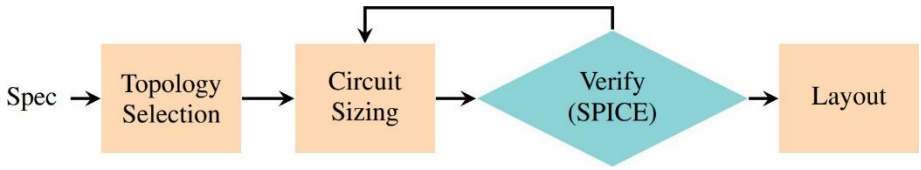

Figure 2.10 – Analog circuit pre-layout design flow[21]

This approach has 3 main variables: $X = x_1, x_2, ..., x_n$, which refers to a finite set of sizing variables; $D = D_1, D_2, ..., D_N$, $D_i = b_1, b_2, ..., b_I$, which corresponds to a non-empty domain, namely the design space; The final variable is $C = C_1, C_2, ..., C_k, C_j = (t_j, r_j)$, represents the constraints to which the problem is subjected, t_j being the constraint scope and r_j being the relation over the variables in thescope.

Given the situation, the preferred algorithm is local search due to three main advantages, being:

1. Faster environment adaptation
2. Model-based agents with supervised learning
3. Easier implementation and convergence

All these virtues combined act as a model-based approach, which aims to imitate the behaviour ofa SPICE simulator. A feed-forward network f_{NN} (X, θ) with three layers can be used as a SPICE function approximation: $\hat{y} = f_{NN}(X, \theta) \approx S_{PICE}(X)$. The loss function is obtained by MSE.

Every search begins with a random exploration in the design space. The next step involves selecting the most optimal point as the local area D_L. The idea behind it is that, upon modelling the local landscape, a candidate solution can be selected. This selection is made in the local domain and is based on the expected value computed with the measurements in $\hat{y}$. Local area's properties are granted by the trust region method (TRM).

The fundamental part of this work lies in the trust region method (TRM). This method characterizes a trust region radius δr_i, which is iteration dependent. The model ($V_{alue} \circ f_{NN}$) is trusted to be suitable representation of the objective function ($V_{alue} S_{pice}$). Upon each iteration, the trust region sub-problem is solved by a trust region algorithm to obtain a vector of optimal trial steps $d^{*(i)}$. Afterwards, a ratio ρ^i between the estimated reduction as well as the actual reduction is calculated. This ratio is a criterion used to classify the trial step. For instance, if the ratio is non-significant, that will lead to a trial point rejection.

The final step consists of updating the radius taking the ratio ρ^i into account. In case the NN can closely approximate the objective function $V_{alue} \circ S_{pice}$, the trust region is expanded, otherwise, the trust region is sure to diminish.

As for reward computation, a value function is created with the goal for estimating the merit of circuit measurement sets (after a SPICE simulation is run). The value function acts as a guide since it points the direction of the next move. This differs from actor-critic methods since values are not a part of training. The utilized method is relatively simple. It is based on the sum of normalized measurements so that no extra information is gathered. When analysing the trade-off between constraints, there is a concept which could be implemented. It lies in the premise of a second-stage value function, something that can be implemented with the purpose of assigning importance to each measurement. These actions can be performed once the agent enters an optimal local area.

Experiments were conducted on both academic and industrial settings. The agents are first tested on a two-stage OpAmp with BSIM 45/22mm processes simulated on an open sourced NGSPICE simulatordeveloped by UC Berkeley. The first step is to benchmark the proposed method with several different baseline models, which include random search, customized BO, Advantage Actor-Critic (A2C) [22], Proximal Policy Optimization (PPO) [23], and Trust-Region Policy Optimization (TRPO) [24], implemented by Stable-Baselines [25].

All these agents follow the same observation design as AutoCkt [19] and use the same reward functionas the proposed agents. Every experiment has a 10000-step limitation. As for the customized BO and the proposed model-based agents, 100 experiments are run to prove stability. On the other hand, only 10 are executed for agents in stable baselines since the experiments take an excessive amount of timeto complete (about a month). The results are visible in the table below:

Table 2.5 – Performance of agents in 45nm two-stage OpAmp

	Success-rate	**Average iterations**
Random search	100%	8565
Customized BO	100%	330
A2C	90%	34797
PPO	40%	31503
TRPO	20%	16350
TRM	100%	36

These experiments prove that random search is a strong baseline in which model-free agents such as A2C, PPO, TRPO fail to reach its performance level. Model-based agents are reported to meet all specifications on 36 steps (on average), indicating stable search.

Since it is not enough to only find a set of sizes in a single condition, PVT exploration strategies are tested on the proposed method. The tests are performed by using two-stage OpAmp with BSIM 22nm process. Random search fails to accomplish the task, as opposed to other strategies, which finish 100% of the time. One compelling discovery is that the initial condition does not influence the result, suggesting that progressive search is not sensitive to the initial condition.

Summing up, the generalizable search framework which is proposed adopts a new approach in the trust-region method, being trained with supervised learning. This results in fast design space adaptation. In addition, a PVT exploration strategy is suggested to deal with different working conditions not considered in previous works.

2.2.6 RobustAnalog: Fast Variation-Aware Analog Circuit Design Via Multi-task RL (2022)

Due to various process, voltage, and temperature (PVT) variations from chip manufacturing, analog circuits inevitably suffer from performance degradation. To tackle this issue, [46] presents RobustAnalog as an efficient variation-aware optimization framework for automatic analog circuit design. One of its primary goals is to reduce simulation cost to design a robust analog circuit not susceptible to variations. The proposed method outperforms methods such as Evolutionary strategy (ES), Bayesian Optimisation (BO) and DDPG and even reduces the simulation cost.

When in the presence of any circuit topology, a search is conducted with the objective of finding a circuit sizing vector which can satisfy the constraints across all variations. The problem can, therefore, be formulated as a constraint satisfaction problem. The prominent goal is to find a sizing vector which satisfies any constraints under any corner task. There is an overview of the proposed framework in Figure 2.11.

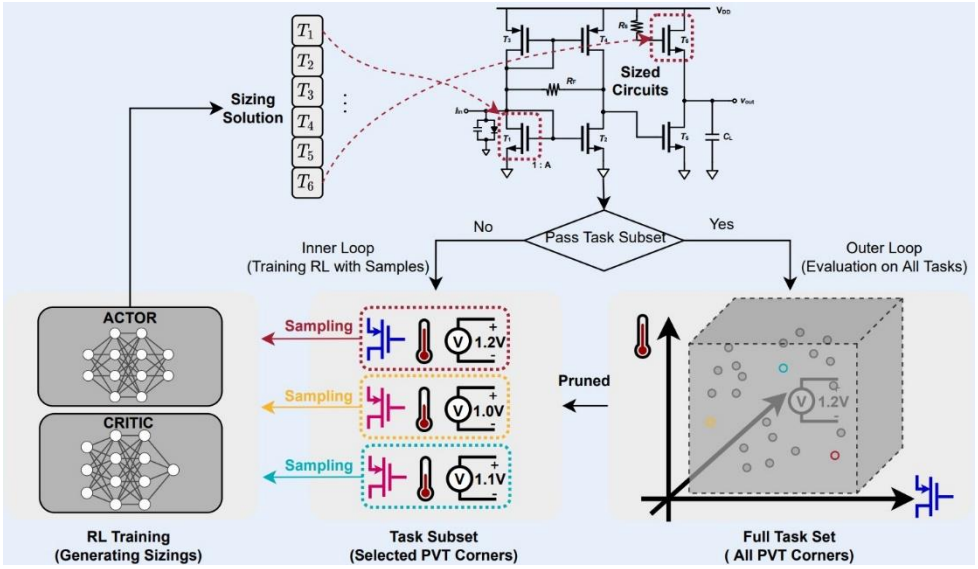

Figure 2.11 – RobustAnalog Overview: (1) A pruned task subset is generated from the full task set (2) Multi-task RL agent is trained on task subset (3) Training continues until the produced sizing can achieve training tasks. Then the sizing is evaluated on the full set. If it passes all the tasks, RobustAnalog returns the result [20]

The process for each iteration goes as follows:

1. RobustAnalog selects a new task subset from all PVT corner tasks;
2. The RL agent generates actions and passes them to each environment in the task subset;
3. The environment denormalizes actions and turns them into actual circuit sizing, followed by a refinement.
4. Circuit Simulation
5. The agent obtains the rewards from corner-specific environments. The actor and critic networksare optimized with PCGrad technique
6. The sizing solution is tested in case all tasks in the subset are passed.

In the meantime, the actor-critic model's weights as well as replay buffers are saved for the agent to inherit once it reaches the next iteration. After going over the problem definition as well as a framework overview, the section that follows involves multitask RL training. Multi-task RL is a training paradigm in which agents are trained using samples from multiple tasks simultaneously. In this case, a multitask agent is created.

This agent's critic can predict the value of task-conditioned action-state pairs. The actor model, on the other hand, is set to be task agnostic. This is due to its target is to look for sizing that passes all tasks. The PVT information is embedded in the states ($s = (p, v, t)$). In this expression, p represents the one- hot representation of component type whereas v is normalised voltage value and t is the normalised temperature value. The reward can be expressed as the measurement of the relative distance betweenthe current performance metrics and the design targets. It is represented by the expression below:

$$f = \begin{cases} r & x < 0 \\ 0.2 & x \geq 0 \end{cases} \tag{2.5}$$

$$r = \sum_{i=1}^{M} min\{\frac{m_i - m_i'}{m_i + m_i'}\}, 0 \tag{2.6}$$

The one thing designers usually prioritize is fulfilling requirements in a short time, hence the lack of over-optimization. As far as the action space is concerned, the action vector is composed by a set of values that correspond to the sizing parameters relative to each circuit.

As for the training, the environment includes the circuit, the simulator and PVT information. Every time the environment is queried, the circuit is stimulated, and it ends up returning the performance with the desired PVT information. Replay buffers proceed to store agent-environment interactions in the formof (s, a, r, x_i) (s-state, a-action, r-reward, x_i-corner task ID). The critic network takes the state, the action, and the corner task ID (s, a, x_i) as input, predicting the corresponding value for the current corner task. The task ID is removed from the inputs of the actor neural network. One aspect that distinguishesthe method from single-task setting is the sampling of a stratified batch from buffers every time as well as generating task-specific losses. As for the the optimization strategy, the used one is PCGrad [40].

Multitask training has been crucial in improving efficiency; however, it is possible to reduce the number of simulations even further by selecting a small-sized training task subset. To improve optimisation,the NN-based RL agent is incrementally trained due to its capability of inheriting weights from last cycleas well as transferability.

Contrary to what human designers usually do, which is guessing the worst-case scenario and then work from that point on, the small batch is chosen from a large batch by sampling in a random fashion. Worst-case scenarios are perceptibly identified as the ones with the lowest reward. The reward value, scalarized from a multi-dimensional performance metric vector, lacks the information to distinguish different low-performance corners. In order to tackle this issue, the corners are clustered based on their multi-dimensional metric values. Thus, inside the cluster, it is possible to get a clearer view of the corner's performance. The proposed task space pruning is divided into three steps, being:

1. Simulation of the optimized on all corner tests to get the performance distribution to each of them
2. Corner division into different clusters based on the performance metrics
3. Selection of the corner with the lowest reward in each cluster as one of the training tasks for thenext iteration

The present method is tested on three real-world analog/mixed signal circuits. The first one is a Two-stage OTA, the second a Folded-Cascode OTA and the last one is a strongARM Latch. RobustAnalog is applied to all three circuits. The results are compared to those of Bayesian Optimization (BO), Evolutionary Strategy (ES), and single-task RL algorithm (DDPG). Each method is compared to the others by the average reward. BO, ES, and DDPG improve the average reward until it reaches the value of 0.2. In ES, DDPG, and RobustAnalog, the circuit simulation time is over 95% of the total time.

RL training was done with a batch of 64, a replay buffer size of 1000,and an exploration noise standard deviation of 0.2. The actor and critic are both a 4-layer MLP implemented in PyTorch, and the evaluation is performed every 10 training steps.

Considering all three circuit benchmarks, RobustAnalog achieves the smallest simulation cost to accomplish all corner tasks. As far as simulation costs go, RobustAnalog outperforms the other methods by a significant amount. The reductions in simulation cost are roughly 26 times in Two-Stage OTA, 30 times in storngARM Latch and 14 times in Folded-Cascode OTA. RobustAnalog shows a very significant improvement in efficiency. Figure 2.12 shows the learning curves for each method. The RobustAnalog method converges faster than the other algorithms for all circuit benchmarks.

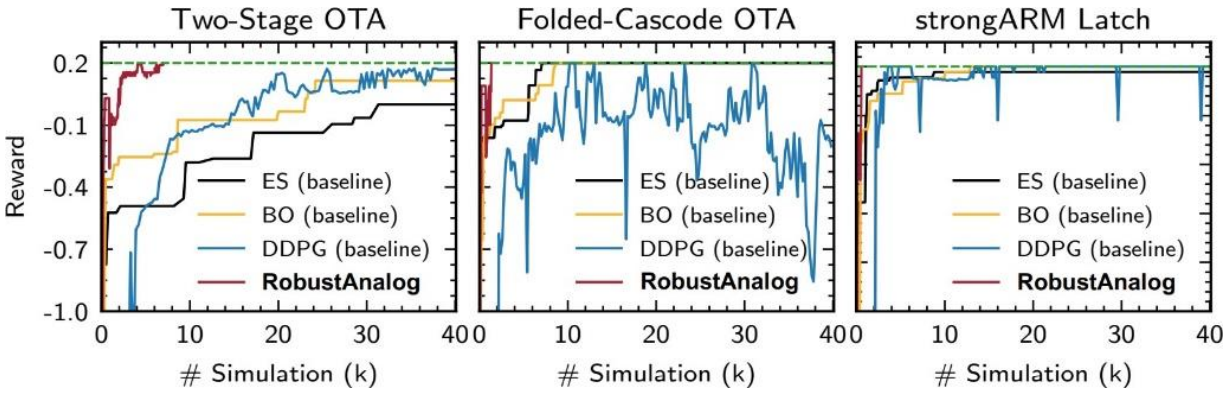

Figure 2.12 – RobustAnalog: Comparing learning curves with baselines (average rewards vs. # simulation).
Reward=0.2 indicates all tasks are passed. RobustAnalog hits the reward of 0.2 significantly faster than the baseline methods on all benchmarks.

RobustAnalog is, therefore, a fast variation-aware optimization framework that is based on multitask RL. Its key property involves the ability to conduct efficient multitask learning with pruned training task space. Hence, it can design circuits effectively for variations. It is shown to significantly reduce simulation cost and scale to many variation cases. RobustAnalog is, thus, a promising approach to drastically shorten the circuit design cycle and reduce the cost.

2.3 Comparative Analysis

Having discussed the most recent attempts to use RL for analog IC sizing, Table 2.6 summarizes the methods' advantages and disadvantages.

The proposed Symbolic Filter applied in [10] not only reduces helps this approach reducesimulation runtime but also offers strong scalability, meaning it can be adapted to other models. One thing that still holds this algorithm back is the long duration of the early period and the runtime not beingoptimal. Although AutoCkt [19] performs very well and has good scalability, it can be time-consuming due tothe extremely large action space.

The Trust-Region method applied in [21] provides a faster environment adaptation, offers an easier implementation and convergence. Its drawbacks include poor extrapolation properties at the beginning of each episode (i.e when the number of samples is low). One recently proposed algorithm, DNN-Opt, explored in [20], offers several advantages. For example, when compared to other algorithms, such as

AutoCkt [19], it is vastly superior in terms of performance. Its ability to size large circuits and efficiency in terms of consumption is also extremely good, making this a trustworthy algorithm

Another example of a proposed algorithm that distinguishes itself is the RobustAnalog[46] [46] since it can conduct efficient multitask learning with pruned task space. The simulation cost is greatly reduced, and this approach proved to effectively design circuits for variations. The CAN Actor-Critic [18] excels in performance as well as improving robustness. Its low variance allows it to perform better that most models. The Actor-Critic associated to the circuit attention network offers a wide range of possibilities. The Critic network in the CAN is able to assess an action and its outcome before it is performed by the actor network, possibly saving complex actions that fail to improve the current solution. Furthermore, its good scalability enables it to transfer knowledge to different topologies as well, which is crucial as to meet market demands.

An illustrative way to describe the way actor-critic works is to imagine the relationship between an infant and its mother. The infant, due to its characteristic curious nature, is bound to explore the environment around it, whether it is by playing with toys, painting the wall, or run around making noise. The infant, in this case, represents the actor. The child's mother (which represents the critic) is bound to watch it and either praise the infant or reprimand it, to help adjusting its behaviour. Eventually, the child learns what behaviours result in praise and tends to have such attitudes in the future, avoidingnegative ones as much as possible.

2.4 Conclusions

In this Chapter, State-of-the-Art for Reinforcement Learning is presented. A series of techniques connected to Reinforcement Learning and their application on analog circuit sizing is described. To evaluate the quality and performance of every technique, each one is discussed in detail. Furthermore, an analysis of both their advantages and drawbacks was performed. Several aspects are considered when making this detailed analysis: state and action space, reward function, how the algorithm generalizes to different situations, and the efficiency and speed for which each algorithm finds a solution. Additionally, throughout several articles, most algorithms were compared, providing a clearer idea of the strengths and weaknesses of each RL method when exposed to the rest of the proposed methods.

Taking all the facts above into consideration and detailed research on the subject, actor-critic is chosen as the preferred RL method to apply to analog circuit sizing. The actor-critic method excels in performance, combining the strengths of policy-based RL and value-based RL, making it the ultimate RL method. This algorithm is divided into the actor and critic networks. The Critic network is extremely useful since it allows the evaluation of an action and whether the outcome will benefit the model before it is performed by the actor network.

Therefore, it possibly prevents complex actions that fail to improve the current solution. In addition, its versatility makes it one of the best algorithms. Therefore, all the previously mentioned reasons contributed to the choice of the actor-critic algorithm as the preferred method for this work.

Table 2.6 – Comparative Analysis between RL methods

Method	State Space	Action Space	Rewards	Advantages	Drawbacks
Policy Gradient				Solves the evaluation metric mis-match during training / test	Unrealistic reward computation; Highvariance
Deep RL [5]	Array containing every circuit parameter	Increment / Decrement in parametervalue	Performance; Attribute weight $k_i \in [0, 1]$ - maximise / $k_i \in [-1, 0[$ - minimise	Symbolic filter - Reduces runtime,strong scalability	early learning period can be long; still not optimal interms of time
Deep Q-Learning				Useful in problems with high-dimensional state space, meaning it can solve more complicatedtasks with lower prior knowledge	Fails when the Q-function (i.e., re-ward function) istoo complex to be learned
CAN Actor-Critic [10]	Transistor sizes $x=[x_1, x_2, ..., x_n]$ in their legal range$[x_L, x_U]$	Transistor size change $a \in \{-1, 0, 1\}^N$	Modified Figure of Metric FOM_R	Improves robustness andperformance, low variance, performsbetter than most models	Can be time-consuming
AutoCkt [11]	Transistor parameters and Design specifications	transistor: width and multiplier; resistor: noof resistorsin seriesand no of resistor inparallel	Cumulative rewards based on design specifications	Great generalisation	Very large action space
DNN-Opt [12]	Optimisation parameters $s_k = x_k$	Change in optimisation parameters $a_k \Delta x_k$	(Not explicit)	Able to size large circuits; Efficient in power consumption	Due to it being so recent, there have not been tests to its scalability
TRM [13]	(Not explicit)	(Not explicit)	Sum of normalised measurements	Faster environment adaptation; Easier implementation and convergence	Can have bad extrapolation properties at thebeginning of theepisode, when onlyfew samples exist
RobustAnalog [37]	$s = (p, v, t)$	Sizing parameters relative to each circuit	Equations 1 and 2	Smallest simulation cost toaccomplish all corner tasks; Very significant improvement in efficiency; High Scalability	Necessity of pruning the task space

3 Deep RL-Based Analog IC Sizing

This chapter is dedicated to the detailed description of implemented work. The first step involves an overview of the agent. Following that, an overview of the agent's network is provided. It is followed by a characterization of how the agent is trained and the way it learns. The latter includes a depiction of the process of action selection, utilized rewards, and how it deals with the exploration exploitation trade-off. The main goal of this work is turning the sizing process of the circuit to an automatic process using the reinforcement learning method Advantage Actor-Critic (A2C).

3.1 Proposed Approach

The Analog IC Design Automation (AIDA) software tools is an automation solution developed and maintained at Instituto de Telecomunicações (IT). AIDA's automatic sizing is based on the NSGA2 multi-objective optimization kernel, accounting for PVT, variability, layout generation and post-layout considerations. The AIDA framework implements an analog IC design flow from circuit-level specifications to a physical layout description, focusing on design optimizing and porting. It uses highly efficient searching methods combined with accurate circuit-level simulation, layout design rules and parasitic extraction engines.

This work, although not directly integrated into AIDA's tool, it does follow its flow providing an alternative Deep RL-based sizing approach. The circuit setup, including netlists (circuit and test benches), circuit parameterization and design variables, and target specifications is the one from AIDA. To that end, the interface with the simulator and measurements processing are wrapped in an environment suitable for RL. An agent is trained to replace the optimizer from Figure 3.1 – Proposed method illustrative diagram, the interface with the simulator and measures is wrapped in an environment suitable for RL, and an agent replaces the optimizer.

3.2 Agent Overview

This section is destined to the description of the agent. The used algorithm is a variation of the Actor-Critic, named Advantage Actor-Critic (A2C) This version of Actor Critic is characterized by the advantage factor. The main purpose of the advantage function is to evaluate how much better an action is when compared to other actions for a particular state as where the value function measures how beneficial that action is for that state. Figure 3.2 depicts the general functioning of the A2C algorithm:

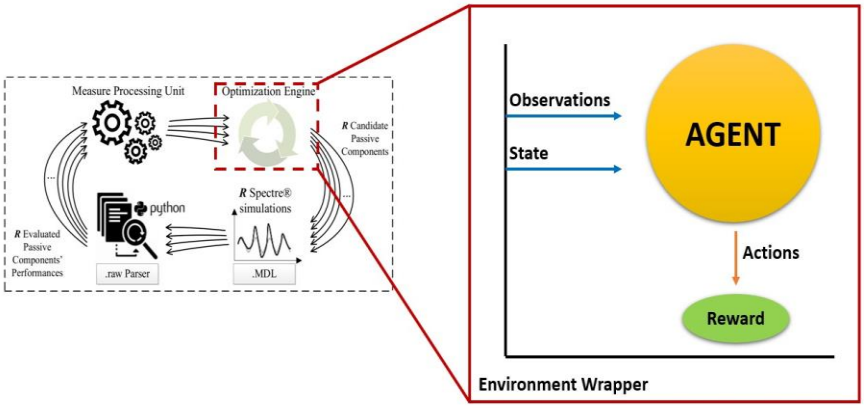

Figure 3.1 – Proposed method illustrative diagram

Algorithm 1 Advantage actor-critic - pseudocode

// Assume parameter vectors θ *and* θ_v
Initialize step counter $t \leftarrow 1$
Initialize episode counter $E \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta)$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t+1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t,\theta_v) & \text{for non-terminal } s_t \text{ //Bootstrap from last state} \end{cases}$
  for $i \in \{t-1,\ldots,t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ: $d\theta \leftarrow d\theta + \nabla_\theta \log \pi(a_i|s_i;\theta)(R - V(s_i;\theta_v)) + \beta_e \partial H(\pi(a_i|s_i;\theta))/\partial\theta$
    Accumulate gradients wrt θ_v: $d\theta_v \leftarrow d\theta_v + \beta_v(R - V(s_i;\theta_v))(\partial V(s_i;\theta_v)/\partial\theta_v)$
  end for
  Perform update of θ using $d\theta$ and of θ_v using $d\theta_v$.
  $E \leftarrow E+1$
until $E > E_{max}$

Figure 3.2 – Advantage Actor-Critic (A2C) - Pseudo Code[41]

Taking that into consideration, it is substantially better to make the model learn the Advantage Values instead of the Q values. This way the action evaluation is based on how good an action is, as well as how better it can become. Another favorable aspect of the advantage function is that it stabilizes the model, reducing the variance of policy networks.

To build an actor-critic method, like any Deep RL framework, it is necessary to specify the following components: state space, action space and reward function. Chapter 3.2.1 describes the states and actions used in this work, as well as the reward function. This is followed by a detailed description of the model's architecture and hyper-parameters. Chapter 3.2.3 describes the process of action selection, providing insight on how the actions are determined based on the current state.

3.2.1 State Space, Action Space and Reward Function

Each state is composed of two different variable sets: design variables and target specifications. The design variables are the variables that are modified according to the defined actions. Each design variable is defined by a tuple, which has three different positions. The first position corresponds to the

minimum size value a design variable can reach whereas the second position is the maximum size value the same variable is able to reach. Finally, the third position is the amount each action alters the size of the component. The target specifications, however, are manually set in the environment wrapper and remain unchanged during the sizing process.

For instance, considering a CMOS circuit Figure 3.3), the state space involves all the sizes of the components that constitute the circuit plus the target constraints manually set for this model. Assuming there are only 2 variables that need to be sized, the state space consists of a vector where the first two positions are destined to the sizing components and the rest is filled with the manually specified target constraints. The target constraints specified for this work are IDD, GBW, DC Gain, Phase Margin and Figure of Merit.

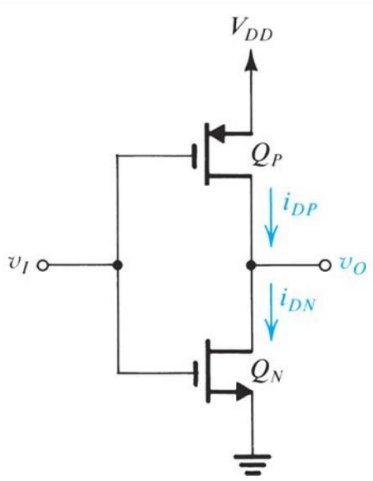

Figure 3.3 – Inverter circuit - CMOS

An action in this framework is an alteration in the state. These changes are only applicable to design variables since the target specifications remain unchanged. Each action is defined by a tuple, which encompasses the design variable that needs to be altered as well as the amount by which it can change (called the “step size”). For this work, the step size defined for each action is either 1 or 10, meaning that an action can increase/decrease by one or ten times the step size of each design variable.

Taking the CMOS circuit in Figure 3.3 as an example once more, if there are two design variables, then the number of possible actions is 4. Each action either increases or decreases the size of the component and since there are 2, there are 4 possible actions. Every time an action is performed, the state suffers an alteration and that results in a change of the specifications that measure the system’s performance. Such alterations are bound to affect the performance of the circuit. Once the target specifications are met, the algorithm considers it has found a solution.

Finally, any RL model needs a reward function. This function is crucial in helping the agent determine which actions are beneficial and which actions are not. Positive rewards are associated with actions that improve the system’s performance whereas negative rewards are associated with actions which worsen it. The RL process is divided into episodes and timesteps. An episode is composed of time steps and each episode in this work is composed of 50 timesteps. Upon each time step, an action is performed, resulting in a reward assignment. If the action brings the circuit performance measures

closer to the target specifications, it means the circuit performance was improved, resulting in a positive reward.

On the other hand, if the action leads the circuit performance away from the target specifications, the assigned reward is negative. The reward for each action is the difference between the target specifications that are not fulfilled and the objective target specifications, similarly to what is defined in AIDA.

$$g_0(x) = \frac{gbw}{35 \times 10^6} \tag{3.1}$$

$$g_1(x) = \frac{pm}{65} \tag{3.2}$$

$$g_2(x) = 1 - \frac{pm}{90} \tag{3.3}$$

An action results in an alteration of the state and, if the agent gets closer to the target specification, it results in a reward increase, otherwise, the reward value decreases. This calculation is performed for every single target constraint (IDD, DC Gain, GBW, Phase Margin, and FoM). The resulting reward for each action is obtained by adding each of these results. If a solution is deemed impossible to achieve, the episode is immediately terminated and a reward of -100 is assigned to the episode. The reward is propagated to the next episode, so the agent has prior knowledge of the actions when the following episode begins.

The reward function according to the state can be defined as:

$$Reward(s) = \begin{cases} -100, & if \sum c_i(s) < -3000 \\ \sum c_i(s), & if -3000 < \sum c_i(s) < 0 \\ 2000, & if \sum c_i(s) \geq 0 \end{cases} \tag{3.4}$$

$$c_i(s) = \begin{cases} g_i(s), & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{3.5}$$

Thus, the agent enters each episode with progressively more knowledge of the implications of each action. The reward for each episode is calculated through adding a portion of the total rewards of the current episode (95%) and a portion of the rewards of the prior episode (5%). Once a feasible solution has been found, the episode terminates and is given a reward of +2000.

3.2.2 Model Structure and Hyper-parameter tuning

The Actor-Critic model is defined as a large Neural Network. Like all ANNs, the proposed network has an input layer, an output layer, and a set of hidden layers. This work makes use of two NNs: the first one employs two common layers, later dividing itself into actor and critic, however, the hidden layers are common to both networks; the second model has no common hidden layers; hence, the actor and critic are trained separately:

The first network is composed as follows:

- 1 input layer
- 2 common hidden layers
- 1 output relative to the actor
- 1 output relative to the critic

To better visualize the first network's architecture, there is a representation in Figure 3.4:

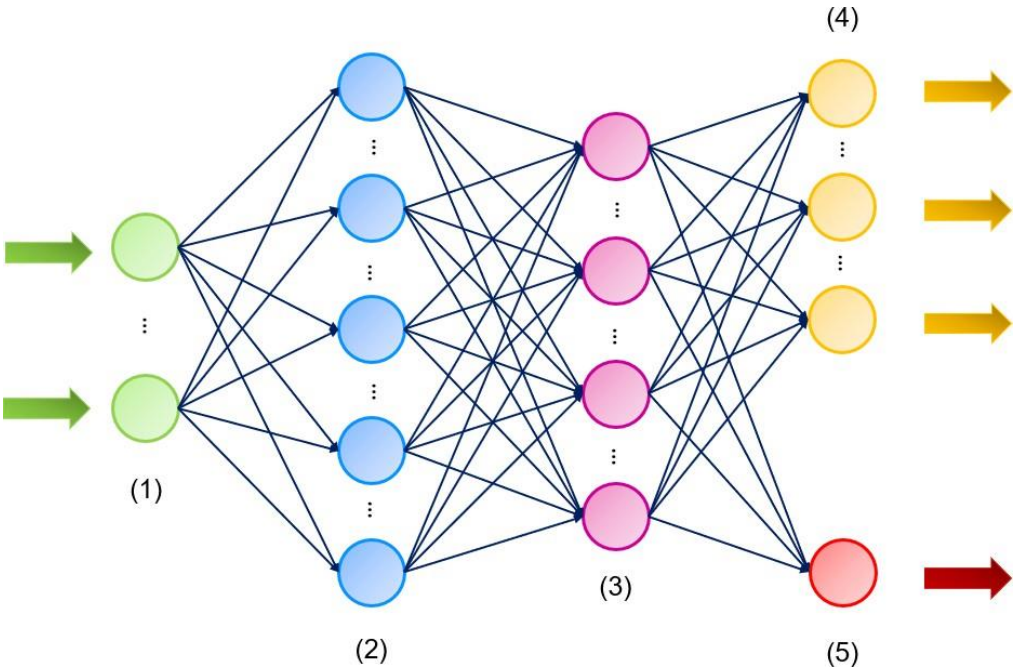

Figure 3.4 – Representation of the model used in this work: (1) Input Layer (2) Common hidden layer 1 Common hidden layer 2 (4) Output layer - Actor (5) Output layer - Critic

The second model is substantially different form the first one, since the actor and critic networks are trained separately:

- 1 input layer
- 2 hidden layers for the actor
- 2 hidden layers for the critic
- 1 output layer relative to the actor
- 1 output layer relative to the critic

Figure 3.5 offers a visual representation of the model's architecture. As it is earlier stated, the prominent difference between both models lies in the architecture, mainly in the position of the hidden layers. The first model uses both hidden layers prior to the divergence between actor and critic. The hidden layers are common to both actor and critic; however, they produce different outputs. In the second case, both networks are separated, which proves, in theory, to be a more difficult task. Nevertheless, both models are tested in Chapter 4 and their performance is evaluated.

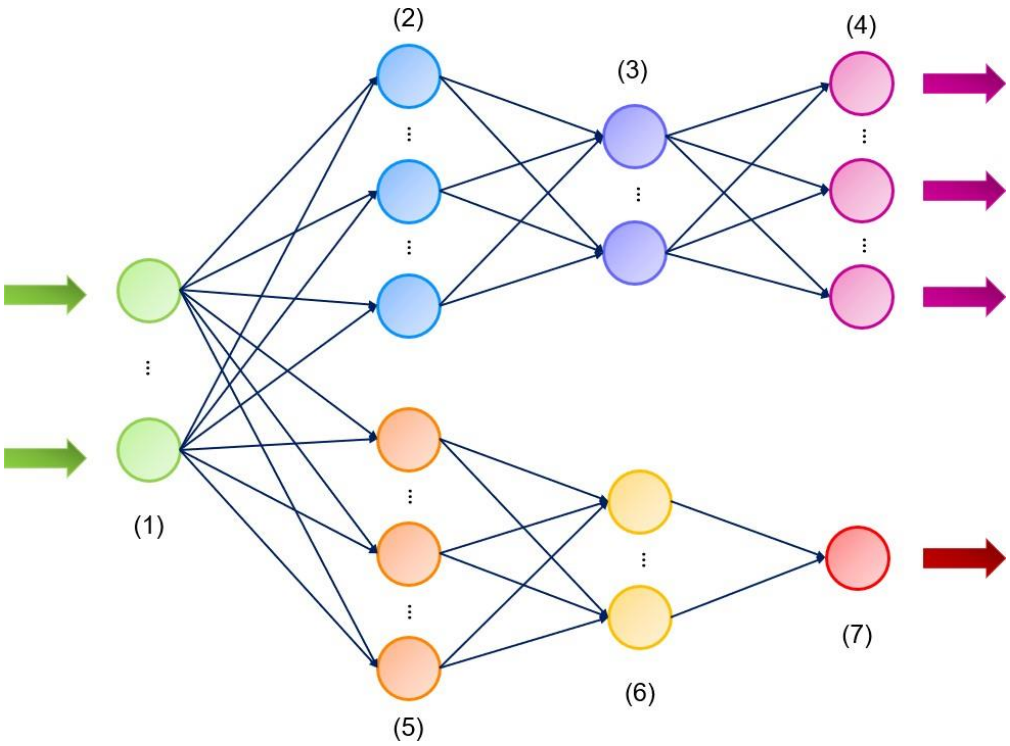

Figure 3.5 – Representation of the model used in this work: (1) Input Layer (2) First hidden layer of the actor network (3) Second hidden layer of the actor network (4) Output layer of the actor network (5) First hidden layer of the critic network (6) Second hidden layer of the critic network (7) Output layer of the critic network (2), (3), (4) Actor Network (5), (6), (7) Critic Network

The input of the model is the initial configuration of the state, which is manually set and common to both models. The actor's output is a tensor vector which corresponds to a probability distribution. The size of the vector depends on the environment since it dictates the number of possible actions. Each action has a probability associated to it. The critic's output is a value function (size 1), which is used to determine how beneficial an action is for a given state.

Model 1: The first hidden layer of the first model (Figure 3.4) has a default number of 1024 neurons and the second one is composed of 512 neurons. The activation function for both hidden layers is the leaky ReLU activation function. After the second hidden layer, the network is divided into actor and critic. The layer prior to the output of the actor is submitted to a sigmoid activation function and the actor's output is then submitted to a SoftMax activation function.

Model 2: The actor and critic network are separated. The actor network (Figure 3.5): 2, 3 and 4 has a first layer composed of 1024 neurons, a second layer formed by 512 neurons. Contrary to the previous model, the first hidden layer is activated through Leaky ReLU activation function whereas the second hidden layer is activated through a sigmoid function. The same is also true for the critic network's first hidden layer, however, the second hidden layer is activated through a Leaky ReLU activation function. The output of the actor is submitted through a SoftMax function. The chosen optimizer is the Stochastic Gradient Descent. One of its parameters is the learning rate. A high value learning rate results in fast updates and average estimates tend to decrease quickly. However, the model is more sensible to fluctuations coming from reward randomness. On the other hand, a low-valued learning rate is much less sensible to randomness and the updates are slow. Average estimates are bound to slowly

decrease, and the "overshoot" risk is reduced, thus increasing the probability of finding the best action. The nature of this problem caused for the implementation of an adaptive learning rate. An adaptive learning rate is a value of learning rate which decreases over time.

To achieve that, it benefits from two main parameters: *gamma* and *step size*. The first parameter controls the decay rate, i.e., how much the learning rate will decrease. The second parameterdefines how frequently the learning rate is updated (example: if *step size*= 2: the learning rate is updated every two episodes). An adaptive learning rate improves performance and require less parameter tuningand less effort in hyperparameter settings. Additionally, they facilitate training in terms of speed. The activation functions chosen for this implementation are: Leaky ReLU, Sigmoid and SoftMax. The sigmoid function is one of the most prominent non-linear activation functions to date. It is an S-shaped function (as shown in Figure 3.6) and has the property of mapping any number into a small range, such as the interval [0, 1]. It is generally used to convert real values that can be interpreted as probability, which is the case for this work.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.6}$$

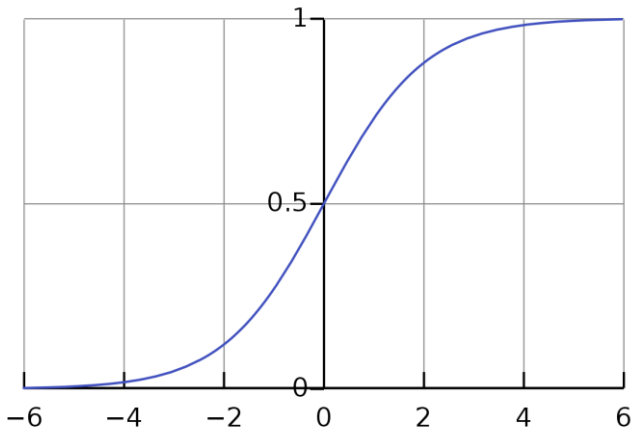

Figure 3.6 – Sigmoid Function

Leaky ReLU acts as an improved version of the ReLU. The classical ReLU activation function has a gradient of 0for negative inputs, which can result in dying ReLU problem for neurons in that region. Leaky ReLU is especially designed to address this situation. Instead of defining the function as 0 for input values lowerthan 0, it is defined as an extremely small linear component of *x*, for example:

$$\sigma(x) = \max(0{,}01x, x) \tag{3.7}$$

Fundamentally, the function returns x if it receives a positive input (identical to ReLU), however, it returns a small value (Figure 3.7) for negative inputs. Hence, it is possible to have an output for negative values. This way, the gradient on the left side becomes a non-zero value, preventing dead neurons. The final activation function worth mentioning is the SoftMax function, which is used for the actor's output.

This function turns a vector of K real values into a vector of K real values that add up to 1. Either the input is a negative, positive or zero, the SoftMax function transforms it into values which fit in the [0, 1] interval. This way, the output can be interpreted as probabilities, which is the goal for the model.

Basically, the system works the following way: if the input value is a negative value, it is turned into a small probability by the SoftMax function, whereas a large input is transformed into a high probability. The SoftMax function proves itself extremely useful since it normalizes the score into a probability distribution. In this work, the Leaky ReLU activation function is applied to every layer except the last hidden layer of the actor network on both models, which is where the sigmoid function is applied. The output layer of the actor (for both models) is subjected to a SoftMax activation function.

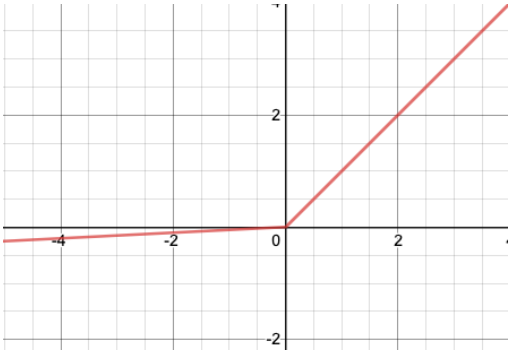

Figure 3.7 – Leaky ReLU Function

3.2.3 Action Selection

Exploration and exploitation are two possible approaches upon facing a decision-making problem. Both approaches have their advantages and drawbacks. Exploitation consists of preferring the decision assumed to be optimal in respect to the observed data. The main goal of this behavior is avoiding poor decisions as much as possible, however, it may prevent the algorithm from finding potentially better solutions.

Exploration involves avoiding decisions that may seem optimal. This process relies on disregarding observed data, which is not considered to be sufficient in truly identifying the best possible option. The main goal is to find a strategy with a favorable trade-off between these decisions. This dilemma is a common problem in most of the data driven decision making process. Currently, there are several different methods which enable the management of this trade-off between exploitationand exploration, among them:

- ϵ-greedy – Chosen action is assumed to be the best in some cases, otherwise explore randomly
- Optimistic initialization - Chooses actions assuming the best for expected rewards until proven contrary
- Upper Confidence Bound - Chooses the action with the highest upper bound estimate of the reward
- Thompson sampling - Chooses an action randomly according to their probabilities to be the best

The exploration-exploitation trade-off implies a decision regarding the current model's state. The split lies in deciding according to the current model (exploitation) or take another decision which can lead to a better outcome (exploration). The actor's target is to produce the action with the best outcome. Hence, the actor's output consists of a probability distribution.

The only usable distribution in this case is the Categorical distribution since the state is a multi-class problem. A categorical distribution is a discrete probability distribution which describes the probability a random variable can take on a value belonging to one of N categories. Each of the categories has a probability associated with it. This type of distribution proves to be advantageous since, like all probability-based exploration methods, enables the agent to randomly sample an action according to its probability to be the best possible action.

Thus, profitable actions have a higher chance of being chosen, although they are not strictly chosen as there is room for exploration. This is due to the existence of a possibility, although lower, that the agent decides on an action which has a lower probability. Because there are episodic updates, if an action that happens to have a lower probability proves profitable, it will be updated accordingly. This way, its chances of being chosen are increased for the following episode.

3.2.4 Circuit Sizing using Deep Reinforcement Learning

Contrary to most ML methods, RL algorithms are not divided into the typical training, validation, and test stages. In model training, a given model is trained to be later tested and validated. This is not the case for this work. The learning process in RL, more specifically, the actor-critic method, is the process of finding a feasible solution given an environment with manually set constraints. Therefore, the principal objective of the learning process is finding a feasible solution for the environment, automating the sizing component of the project.

At first, the neural network's weights are initialized randomly. During the beginning of the process, the quality of the results is especially low because the agent is learning what actions result in better rewards. During the circuit sizing process, the objective is to transform a poor performing agent into an agent that can size the circuit's components, meeting the set target specifications. The agent maintains a policy (actor) π ($a_t|s_t$; θ) and a value function (critic) V (s_t; θ_v). Both policy and value functions are updated after 50 timesteps or in case a terminal state is reached.

$$A(s_t, a_t; \theta, \theta_y) = \sum_{i=0}^{k-1} \left(\gamma^i r_{t+i} + \gamma^k V(s_{t+k;}\, \theta_v\right) - V(s_t;\, \theta_v), \qquad k \leq t_{max} \tag{3.8}$$

The advantage function (Equation 17) is the sum of discounted rewards added to the difference in value functions between states. The Loss function is portrayed in Equation (3.9)

$$L = \log \pi(a_t|s_t;\, \theta)(R_t - A(s_t, a_t, \theta, \theta_v) \tag{3.9}$$

In this expression, the term R_t - $A(s_t, a_t; \theta, \theta_y)$ is relative to the temporal difference term (TD). The R_t term can be expressed as it is stated in Equation (3.10):

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_t + k \tag{3.10}$$

The TD term is multiplied by the probability assigned by the policy for the action at time t. This way, policies which are more certain are more heavily penalized in case they incorrectly estimate the value function. It is important to highlight how it progresses over time. First, it is divided into episodes and time steps. Each episode is composed of time steps and in each time step, an action is performed. The

result of the action proceeds to influence the current state and reward. Afterwards, the episode reward is updated as the last reward of the episode. After each episode is finished, the reward function is updated, leading to an update in the solving condition. Once the reward function is updated, the training process enters in the finishing step.

The finishing step is the final step in an episode, incorporating every update the model needs to progress to the next episode. The final step ensures the model is ready to progress to the following episode and improve, by applying prior knowledge. The present rewards are updated considering the past rewards multiplied by the discounting factor and total episode rewards. The result is used to calculate the advantage function. The advantage function factors how beneficial an action is when compared to other actions for a particular state. Following the advantage function calculation, both policy losses and value losses are calculated. Once they are calculated, the gradients are all set to 0. (This step is important because PyTorch accumulates gradients, compromising the backpropagation and weight adjustment stages). The policy and value losses are added to achieve the total loss, which contributes to updating the weights in the Deep NN. Once the loss is fully calculated, it is possible to perform backpropagation, thus updating the weights and improving the network. The entire process is represented in Figure 3.8

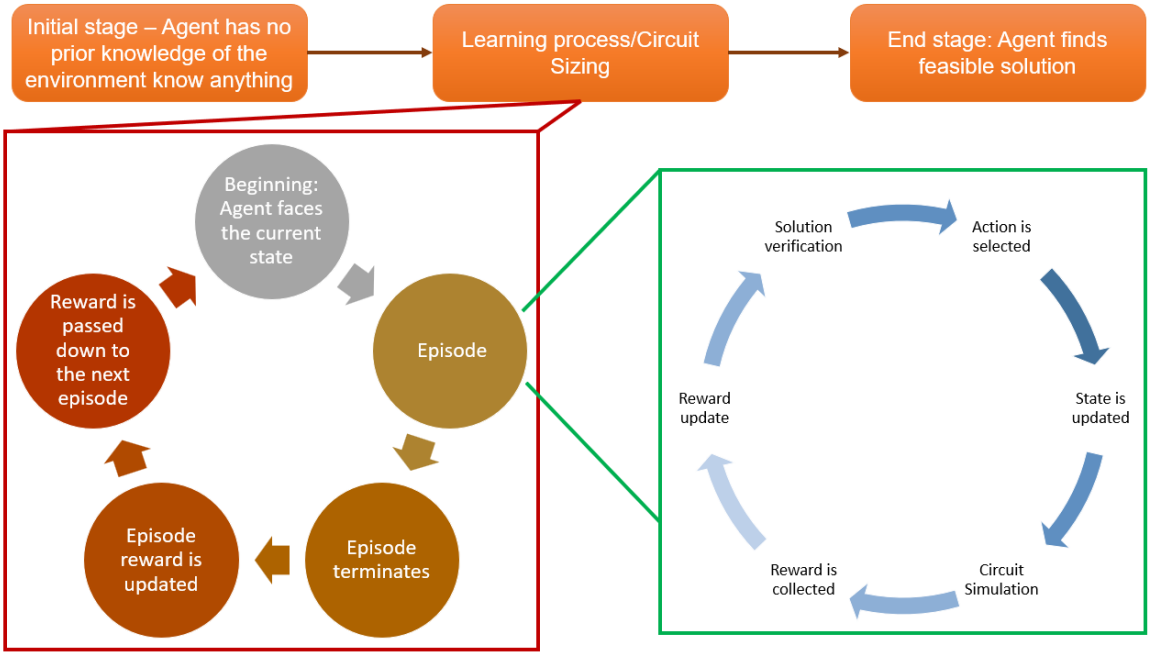

Figure 3.8 – Sizing Process illustrative diagram

3.3 Conclusion

This section describes the work which develops an alternative to analog IC sizing, using deep reinforcement learning. The first section provides an overview of the proposed approach to the analog IC sizing problematic. The implemented work is set to provide an alternative Deep RL-based sizing that follows the flow of AIDA. The interface with the measurements processing and simulator are wrapped in an environment suited for RL. The agent is ultimately trained to replace the optimizer.

The second part of this Chapter provides a detailed description of the agent and how it is implemented in this work. The state space as well as the action space are the first things that need to

be highlighted. Each state is composed of a vector which contains the components that need to be sized as well as the target specifications. Each action increases/decreases the component's size by the amount defined in the ”step” vector. Finally, to determine if the agent is learning and to improve its learning capabilities, a reward function was implemented. if the action improves the results, the reward is a positive value. Contrary, if the action taken for a given state worsens the system, the reward takes a negative value. The main goal is to reach the 0 value. If this value is surpassed, it means the model has found a solution. This is followed by a thorough description of the model's structure.

There are two different structures: the first model has two common hidden layers and only separates the actor and critic networks at the end whereas the second model divides into actor and critic earlier on, resulting in the trainingof two different networks. Finally, a thorough description of the entire sizing process is given. The process is divided into episodes and each episode is composed of timesteps. The sizing process illustrates how each action affects the state as well as the reward and how the agent evolves during its interaction with the environment. Unlike typical training process, the agent in Reinforcement Learning learns as it interacts the environment The learning is divided into episode and in each episode, it learns progressively more about the environment. The end of the process happens when the agent is able to find a feasible solution.

4 Results

This work is implemented in PyTorch, a Python framework. PyTorch is tightly integrated with the Python language. When it comes to debugging, PyTorch offers a wide variety of debugging tools such as a pdb, PyCharm debugger or ipdb. The *nn.Module* is a building block PyTorch provides in order to create complex deep learning architectures. PyTorch is also full of ready to used modules in *torch.nn* package (which are used as a base for the proposed model). The operations are relatively low-level operations. Overall, PyTorch provides useful tools to simplify code and speed up the development of models. The present work is implemented in PyTorch, mainly due to it being clearer and developer friendly. The *torch.nn.Module* is a flexible and useful tool as well. Its accessibility and relatively easier experience in debugging and development is the main reason why PyTorch has been chosen for the development of this work

This Chapter is destined to the presentation of the results derived from the implementation. Every model in this work, both agent and environment, are implemented in Python 3.8.8, using PyTorch as backend. The code is run on an *AMD Ryzen 7 5700U with Radeon Graphics CPU 1.80 GHz* with 16GB of RAM.

4.1 Dataset

For proof of concept, the amplifier using voltage combiners for gain enhancement (VCOTA), presented in [42], is considered as well as the folded cascade amplifier (FCA), used in [43]. Voltage-combiners are typically used in radio frequency, due to their ability to convert fully differentialsignals into a single-ended one, for 50 and 75 Ohm impedance matching. The electrical scheme of a VC is shown in Figure 4.1. It employs a combination of a NMOS in a common-drain configuration and anNMOS in a common-source configuration.

The amplifier topology using voltage combiners proposed in [1] was optimized to maximize the figure of merit (FOM) and maximize de low-frequency gain (GDC). By maximizing the FOM, the power consumption is minimized as well as the gain-bandwidth product (GBW) is maximized, as described by the FOM is in (1), where C_{load} is the load capacity and *IDD* is the current consumption.

$$FOM = \frac{GBW \times C_{load}}{IDD} \left[\frac{MHz \times pF}{mA}\right] \quad (4.1)$$

The circuit schematic is shown in Figure 4.1 and the performance figures are measured using ngspice. The extracted measures that can be used to define objectives as constraints are listed and described in Table 4.1.

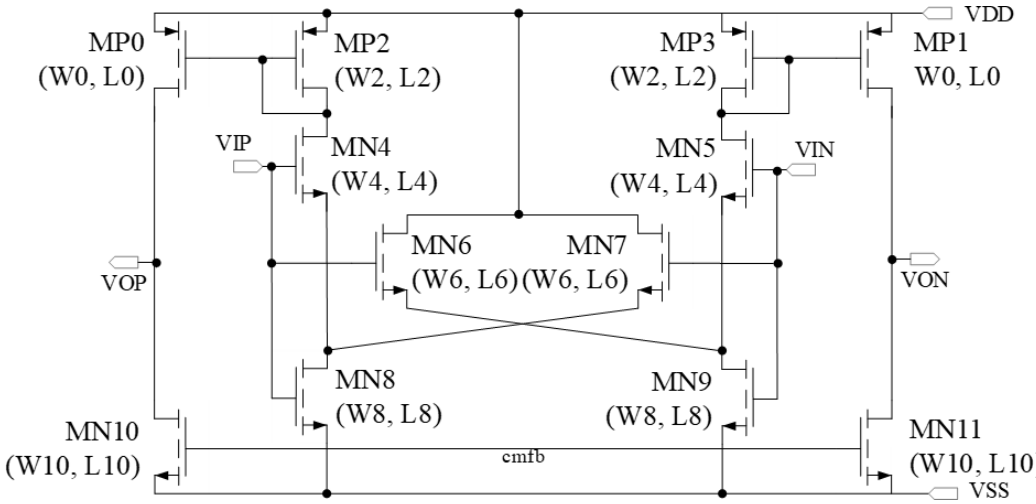

Figure 4.1 – Circuit schematic showing the devices and corresponding design variables (channel width:W's, and channel length: L's)

Table 4.1 - VCOTA performance figures measures in the circuit simulation

ID	Units	Description
IDD	A	Current Consumption
GDC	dB	Low-Frequency Gain
GBW	Hz	Unity Gain Frequency
PM	degree	Phase Margin
FOM	MHz * pF / mA	Figure of Merit
OVP^n	mV	Overdrive Voltages of the PMOS nth Device (VTH-VGS)
OVN^n	mV	Overdrive Voltages of the NMOS nth Device (VGS-VTH)
DP^n	mV	Saturation Margin of the PMOS nth Device (VDSat-VDS)
DN^n	mV	Saturation Margin of the NMOS nth Device (VDS-VDSat)

For the setup of the VCOTA, the devices' sizes that constitute the optimization variables were the width, length and the number of fingers of all the MOS devices, and the dimension ranges are presented in Table **4.2**.

Table 4.2 - **VCOTA variables and ranges.**

Variable (Unit)	Min.	Grid Unit	Max.
w8, w6, w4, w10, w1, w0, l8 (µm)	1	1	100
l6, l4, l10, l1, l0 (µm)	0.34	0.1	10
nf8, nf6, nf4, nf10, nf1, nf0 (µm)	1	1	8

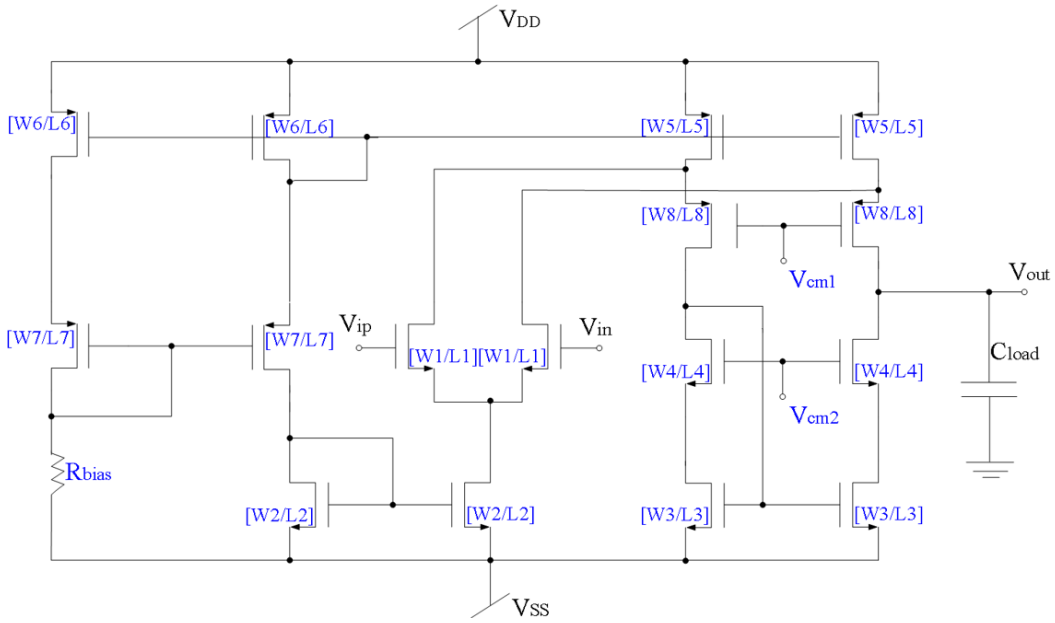

Figure 4.2 – Circuit schematic for the FCA showing the devices and corresponding design variables (channel width: W's, and channel length: L's, bias voltages ($V_{cm1,2}$), and the bias resistor (R_{bias})

The principal goal of this architecture is testing how the actor-critic method performs in the sizing of the components, while keeping the performance of the circuit to the highest level and fulfilling the established constraints. The measures are extracted from the circuit simulation and can be used to define objectives and constraints are listed and described in Table 4.3.

Table 4.3 – FCA's performance figures.

ID	Units	Description
IDD	A	Current Consumption
GDC	dB	Low-Frequency Gain
GBW	Hz	Unity Gain Frequency
PM	degree	Phase Margin
NO	V / V	Noise RMS
SN	V / √Hz	Noise Density
Sr		Slew Rate
Voff	V	Offset Voltage
PSRR	V	Power Supply Rejection Ratio
OVP^n	mV	Overdrive Voltages of the PMOS nth Device (VTH-VGS)
OVN^n	mV	Overdrive Voltages of the NMOS nth Device (VGS-VTH)
DP^n	mV	Saturation Margin of the PMOS nth Device (VDSat-VDS)
DN^n	mV	Saturation Margin of the NMOS nth Device (VDS-VDSat)

The design variables are the width, length, number of fingers and number of rows of the MOS devices, and the length and number of fingers of the MOM capacitor. The variable ranges considered are indicated in Table 4.4.

Table 4.4 – FCA's optimization variables and ranges.

Variable (Unit)	Min.	Grid	Unit Max.
wp5, wp3, wp1, wp0 (µm)	1	0.1	100
wn7, wn5, wn3, wn2, wn1 (µm)	1	0.1	100
lp1, lp0, ln7, ln5, ln3, ln1(µm)	0.34	0.1	10
nfp5 nfp3, nfp1, nfp0	1	1	8
nfn7, nfn5, nfn3, nfn2, nfn1	1	1	8

4.2 Loss Function

A loss function is of extreme importance in determining the accuracy of any model. Therefore, to evaluate the performance of the actor-critic network, the model kept track of its losses.

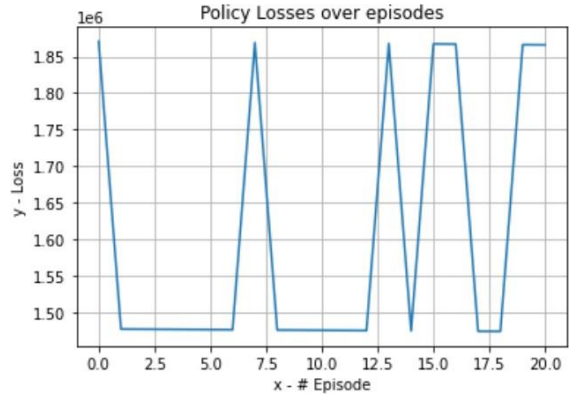

Figure 4.3 – Policy Losses during the training process - VCOTA topology

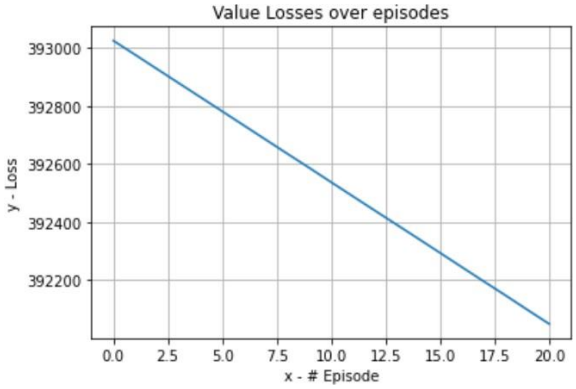

Figure 4.4 – Value Losses during the training process - VCOTA topology

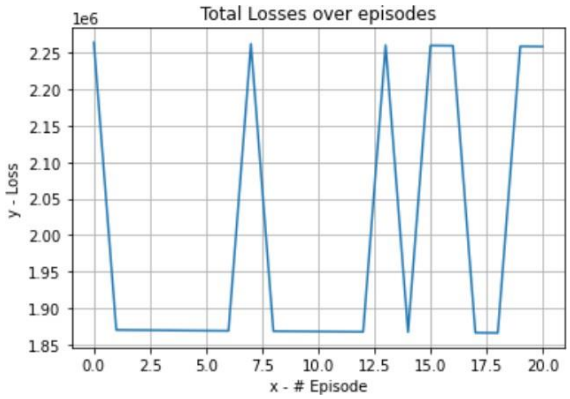

Figure 4.5 – Total Losses during the training process - VCOTA topology

Generally, the tendency is for the loss value to increase during the initial episodes. The reason for this kind of behavior lies in the exploration stage. The agent is exploring different options which are sub-optimal, resulting in a high error function due to the difference between the actual solution and the solution the agent found. Over time, the agent learns what solutions are beneficial and adapts to achieve them, minimizing the loss function. Therefore, although there is an increase in the beginningof the training process, the tendency is for the loss function to decrease after that, as the agent learns and adapts.

However, that is not the case in Figure 4.3, Figure 4.4, and Figure 4.5. It is important to refer that the actor-criticis not a supervised learning method, i.e., when it is learning, it does not know before-hand whether or not the output is close to the solution. Without possessing this kind of knowledge in advance, it is difficult for the model to estimate since the outcome of the actions varies from state to state and different actions have different rewards depending on the current state.

Taking all the above into consideration, it is very complicated for the actor-critic network to predict the outcome of every action for every state, it only knows which actions are beneficial, and therefore, the predictions it makes are off-target, which is a characteristic behavior of unsupervised learning.

4.3 VCCOTA

The selection of the best solutions is organized by targets specifications. Each target has different specification, which needs to be met for the agent to find a valid solution. To test the first circuit (VCOTA), various target constraints are specified. The circuit performances which are considered as a way of evaluating the performance of the RL algorithm are: DC Gain, I_{DD}, GBW, Phase Margin (PM) and figure of merit (FoM). Their ranges of values are shown in the table below:

Table 4.5 – VCOTA: Table displays constraints for each Target

	IDD (µA)	GBW (MHz)	DC Gain (dB)	PM (º)
Target 0:	< 350	> 35	> 50	45 < PM < 90
Target 1:	< 300	> 40	> 40	45 < PM < 90
Target 2:	< 700	> 120	> 50	45 < PM < 90
Target 3:	< 210	> 25	> 40	45 < PM < 90
Target 4:	< 130	> 2	> 40	45 < PM < 90

Target 0 is notably straightforward since the pre-determined conditions are feasible and within reasonably standard values. On the other hand, the constraints on Target 1 are more complicated to fulfil. The I_{DD} minimum limit value is lower than that of Target 0, which, when conjugated to the maximum limit set for GBW, make these constraints more complicated to meet. Target 2 aims to set a higher GBWvalue by sacrificing the maximum limit of I_{DD}. Targets 3 and 4 are designed with the goal of limiting the maximum I_{DD} value while not sacrificing DC Gain in the process. The main difference between Target 3 and Target 4 is that target 4 handles the trade off by lowering the lower limit of GBW while Target 3 does aims to achieve a lower I_{DD} value while keeping the GBW at relatively standard levels. The results obtained using the first model for each target are presented below:

Table 4.6 – Model 1 - Performance results

Seed	# Episodes solved		IDD (µA)	DC Gain (dB)	GBW (MHz)	PM (º)	FoM (MHz×pF/mA)
12	4	Target 0	347,2	55,259	49,45	58,966	854,681
72	50		347,0	58,820	45,57	49,165	787,824
44	124		308,9	58,437	43,67	46,189	848,192
56	70		339,5	55,064	45,72	74,431	807,966
80	21		345,9	54,355	62,64	62,895	1086,436
21	15		261,8	58,576	41,28	49,116	946,052
13	5		345,9	56,154	46,92	55,420	813,982
Average	41,3		328,0	56,666	47,89	56,597	877,876
Std. Dev	43,88		32,26	1,896	6,98	9,868	105,4
12	34	Target 1	285,6	53,750	42,24	79,644	887,283
15	46		260,4	58,449	42,72	48,024	984,351
16	77		298,7	55,889	49,46	45,718	993,533
19	57		279,8	53,053	45,45	46,475	974,619
24	32		297,5	53,412	47,71	58,052	962,313
Average	49,2		284,4	54,911	45,52	55,582	960,420
Std. Dev	18,51		15,59	2,265	3,118	14,34	42,50
12	34	Target 2	615,0	48,429	143,8	73,789	1403,415
14	184		666,4	53,370	124,4	46,399	1120,308
16	201		679,9	52,370	130,7	62,829	1153,398
20	268		593,1	50,920	124,3	63,475	1257,840
80	209		559,1	54,380	120,4	57,045	1292,535
Average	179,2		622,7	51,894	128,7	60,707	1245,499
Std. Dev	87,11		50,39	2,320	9,207	10,02	113,4
12	34	Target 3	209,9	55,771	34,44	72,784	984,773
15	247		207,1	52,864	35,52	63,396	1028,833
24	54		187,2	48,983	26,04	69,794	834,732
21	143		207,1	57,824	30,16	63,036	873,573
22	49		202,0	47,200	26,33	46,743	782,190
Average	105,4		202,6	52,528	30,50	63,150	900,820
Std. Dev	89,98		9,108	4,462	4,419	10,08	103,2
12	39	Target 4	117,6	51,068	4,696	89,493	239,572
15	5		128,7	46,519	2,201	88,901	102,616
22	24		129,2	52,674	15,54	53,254	721,798
24	41		122,9	45,109	4,265	86,979	208,175
21	143		108,9	54,185	4,591	88,005	252,906
Average	50,4		121,5	49,91	6,258	81,33	305,013
Std. Dev	53,74		8,451	3,931	5,286	15,72	24,03

For the VCOTA topology, the results show that the actor-critic network learned the design patterns and found solutions in few episodes. Moreover, it successfully finds solutions within difficult constraints. Given the FoM in every result, it indicates that the solutions it found are efficient, since the FoM value is over 850 in most cases. The FoM is calculated based on Equation (3.4), indicating that the solutions are efficient. For target 0, 1, 2 and 3, the average FoM is over 850, which indicates quality in the solutions that the algorithm found. However, for target 4, the average value of FoM is slightly above 300.

This value isconsiderably low when compared to the other targets. Target 4 compromises GBW in order to achieve alower and feasible I_D/D value. Therefore, since GBW is sub optimal, that change will reflect on the FoM value, resulting in a poor FoM for Target 4. The second model produced the following results:

Table 4.7 – Model 2 - Performance results

Seed	# Episodes solved		IDD (µA)	DC Gain (dB)	GBW (MHz)	PM (º)	FoM (MHz×pF/mA)
12	15	Target 0	282,1	50,073	37,26	75,367	792,401
72	4		325,6	58,129	45,96	60,478	846,799
44	15		336,4	58,144	48,12	58,541	858,187
56	27		314,7	56,318	37,02	77,097	705,892
80	23		338,3	51,148	47,62	73,330	844,450
21	14		260,0	58,465	42,65	48,043	984,204
13	32		317,8	58,128	37,79	49,306	713,579
Average	18,6		310,7	55,772	42,35	63,166	820,787
Std. Dev	9,4		29,16	3,609	4,987	12,22	88,512
12	91	Target 1	286,5	50,966	45,46	67,870	951,973
14	639		280,1	53,292	45,05	46,934	965,030
16	36		298,8	58,106	43,00	57,185	863,319
20	43		295,1	53,818	63,35	58,740	1287,919
80	216		299,8	52,984	44,42	86,082	888,991
Average	205		292,1	53,833	48,25	63,362	991,446
Std. Dev	253,1		8,496	2,621	8,489	14,71	171,1
12	72	Target 2	678,6	55,624	121,1	68,230	1070,738
15	431		651,4	54,481	133,1	48,729	1225,866
16	251		689,8	50,249	129,9	69,897	1130,191
19	231		593,1	50,948	124,2	52,838	1256,638
24	318		660,0	48,621	142,9	80,245	1298,823
Average	260,6		654,6	51,985	130,2	63,988	1196,451
Std. Dev	131,2		37,55	2,953	8,476	12,98	93,76
12	353	Target 3	197,3	55,707	26,79	61,690	814,704
15	272		204,3	45,247	29,24	74,400	858,994
24	67		189,5	52,866	25,86	47,798	818,726
21	318		200,4	57,605	26,99	54,492	807,975
22	31		176,8	56,651	29,02	66,046	984,639
Average	208,2		193,7	53,615	27,58	60,885	857,007
Std. Dev	148,7		10,87	5,002	1,480	10,27	74,08
12	353	Target 4	127,6	50,243	4,672	86,658	219,670
15	87		113,4	43,590	5,345	85,781	282,829
22	31		121,3	46,415	10,76	74,667	532,363
24	30		129,5	42,906	2,835	87,062	131,367
21	485		121,4	56,448	1,169	84,976	577,780
Average	197,2		122,6	47,920	7,061	83,829	348,802
Std. Dev	209,1		6,335	5,574	3,925	5,185	196,5

The results taken from the second model are like those taken from the first model. It is important to remember that the second model has the critic and actor network separated from each other right at

the beginning of the network. Upon thorough examination it is possible to arrive to some conclusions and compare it to the values obtained from the first model.

Before analyzing the results, it is important to clarify that every condition was maintained, only the structure of the model changed.

The number of episodes necessary to reach a valid solution for target 0 decreases on average when compared to its model counterpart (model 1). However, for the remaining Target specifications, the secondmodel needs on average more time to reach a valid solution. This way, the second model seemsto perform better for simpler solutions whereas the first model outperforms its counterpart for more complex solutions, although the difference is small. One aspect worth mentioning is that the number ofepisodes needed to reach a solution varies greatly depending on the utilized initialization value, leading to theconclusion that the seed value greatly influences the number of episodes needed to reach a solution. As for the FoM value, all the results are above 800 (except for Target 4, which is expected). There is a slight decrease in average FoM value for Target 0 when compared to the one obtained by model 1,although the difference is minimal. This observation is also applicable for targets 1 and 3. The differencebetween these values is not significant enough. Surprisingly, the average FoM value increases for target 2 and 4, surpassing the one obtained by the first model. It is important to illustrate how the agent evolves and learns over each episode on different occasions. With that goal in mind, several graphs were taken which show how each target specification evolves in an episode. To get a clearer view of it, different episodes in distinct stages of the sizing process are depicted. The first set is taken for one of the first episodes, the second for one for a later stage, and the third set represents how the different target parameters evolve during the episode for which the agent finds a solution. This process was done for both models and topologies.

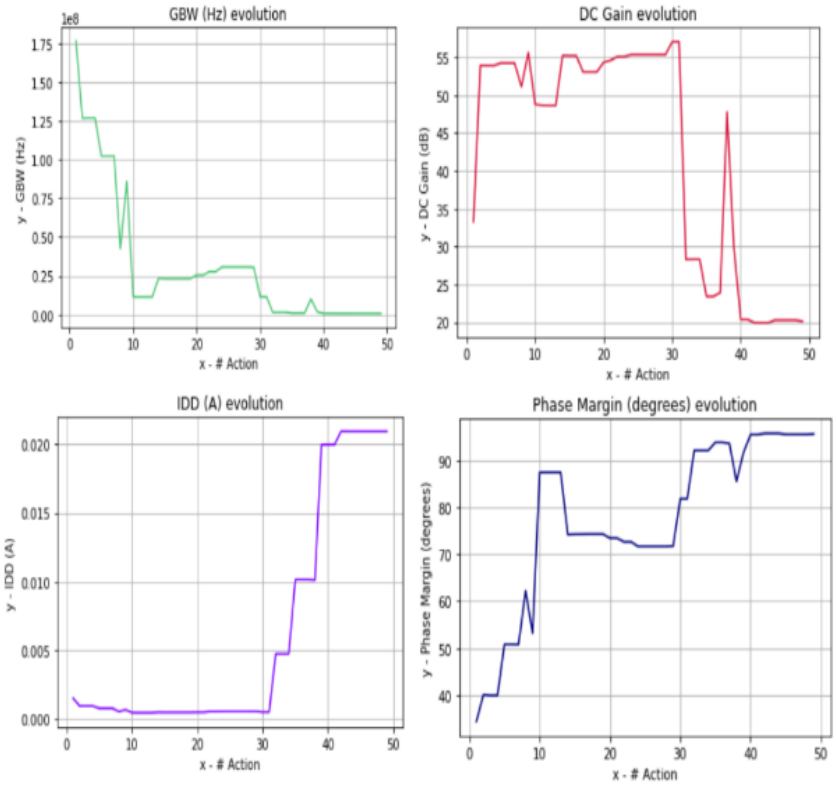

Figure4.6– VCOTA Model 1 – Early Episode

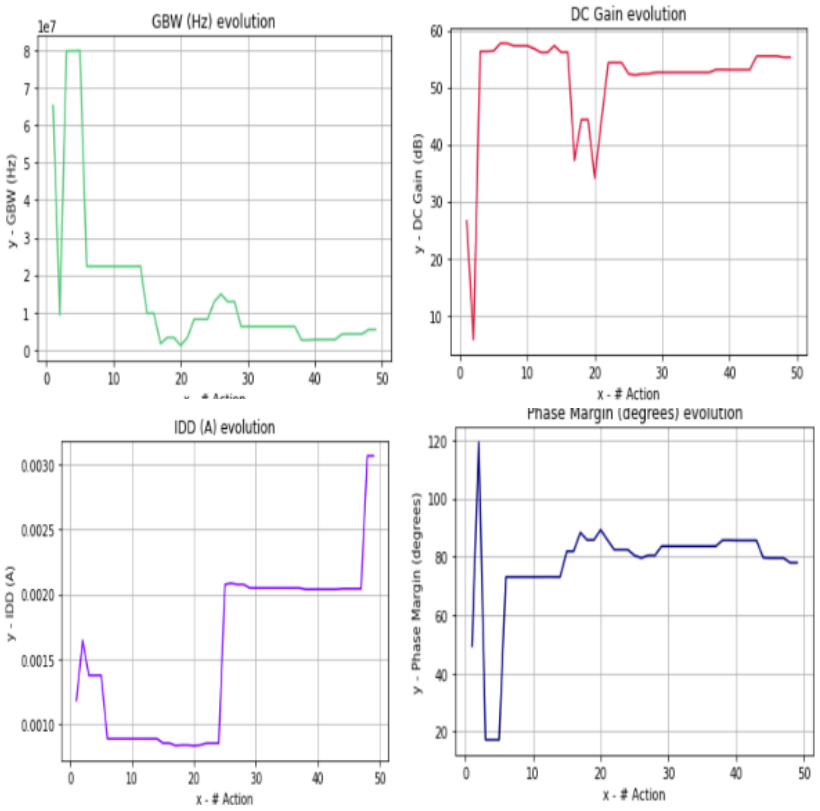

Figure 4.7 – VCOTA Model 1 – Intermediate Episode

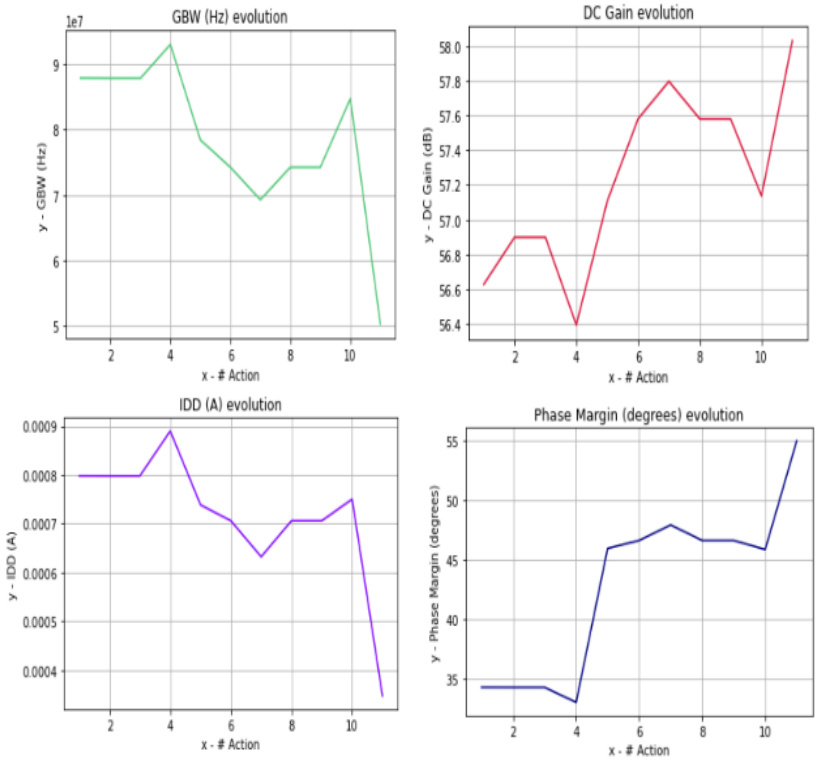

Figure 4.8 –VCOTA Model 1 – End Episode

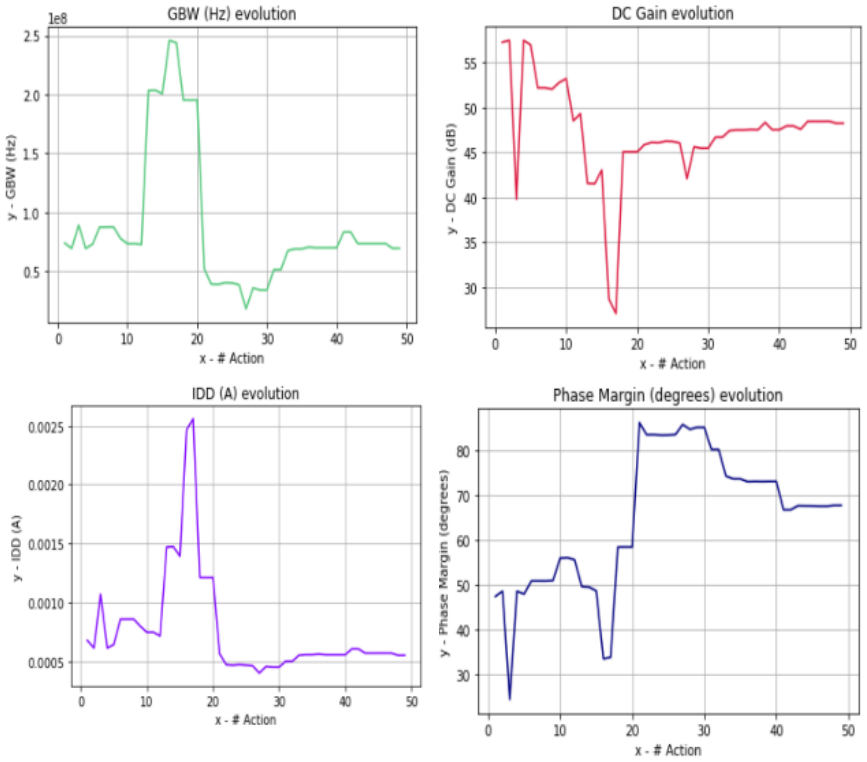

Figure 4.9 –VCOTA Model 2 – Early Episode

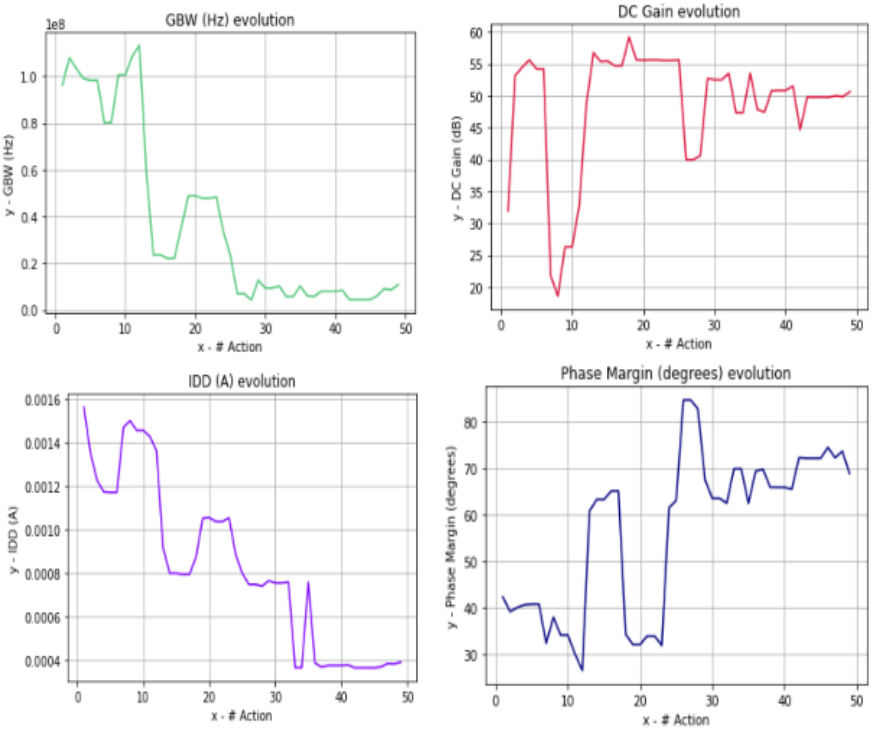

Figure 4.10 – VCOTA Model 2 – Intermediate Episode

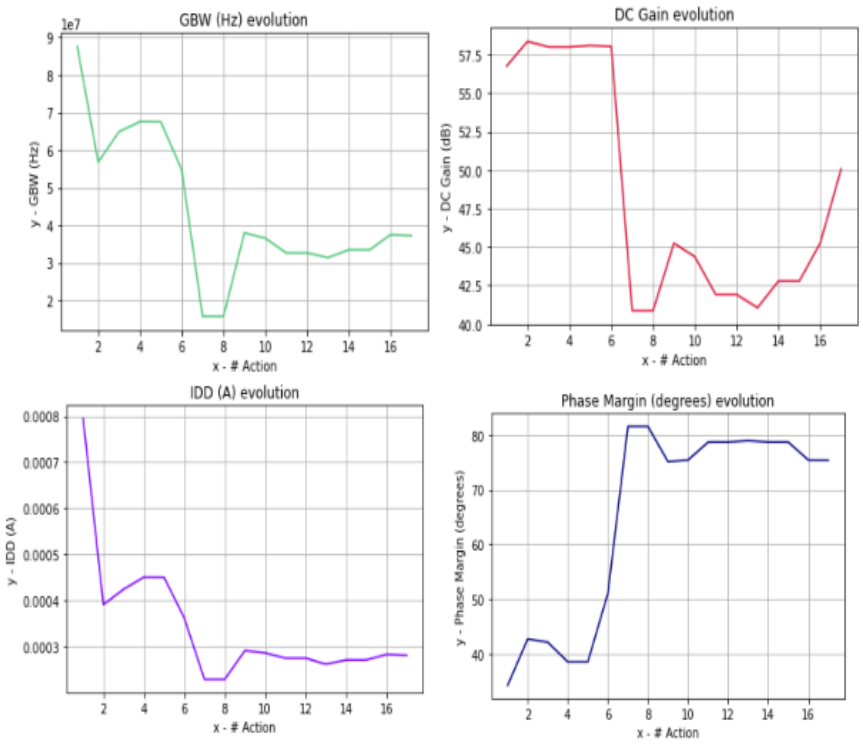

Figure4.11–VCOTA Model 2 – End

Both models obtained successful results, fulfilled the target specifications and within a reasonable time. Overall, the first model performed better than the second one regarding the number of episodesit needed to reach a solution and the average FoM value. However, the difference between both modelsis not large and greatly depends on the seeds, leading to the conclusion that both model structures perform well when applied to the VCOTA topology.

4.4 Folded Cascode

Equivalently to the case presented above, the agent is also tested in the Folded Cascode environment. The main reason behind this is to prove that the agent can adapt to different environments, namelymore complex environments. To assess the performance of the agent, the same procedure is applied to the environment. Like the VCOTA circuit, the circuit performances which are considered as a way of evaluating the performance of the RL algorithm are: DC Gain, I_{DD}, GBW, Phase Margin (PM) and figure of merit (FoM). The ranges of those specifications are specified below:

Table 4.8 – FCA: Table displays constraints for each Target

	IDD (µA)	GBW (MHz)	DC Gain (dB)	PM (º)	FoM (MHz×pF/mA)
Target 1:	< 600	> 45	> 70	45 < x < 90	> 600
Target 2:	< 320	> 35	> 85	45 < x < 90	----------

This time, there are two target specifications: Target 1 and Target 2. The first set of specifications (Target 1) was created as a way of evaluating how the agent handles a several different constraints, balancing an intermediate value of current and GBW, a relative high value of DC Gain. There is also an added constraint relative to the FoM to check if the algorithm can deal with an added constraint. The second target (Target 2) specification aims to conjugate a relatively low current value with a high DC gain value. The agent was tested for the first model and the results are presented below:

Table 4.9 – Model 1, Target 1 - Performance results

Seed	# Episodes solved		IDD (µA)	DC Gain (dB)	GBW (MHz)	PM (º)	FoM (MHz×pF/mA)
12	88	Target 1	428,7	75,204	45,24	79,839	633,170
72	76		351,1	86,502	45,80	58,320	782,683
44	183		371,6	82,526	49,80	78,321	804,090
56	115		316,1	85,910	47,97	74,305	910,535
80	100		388,7	77,283	45,11	65,629	696,321
21	171		330,7	87,116	47,02	61,095	853,099
30	8		282,8	87,397	45,91	69,745	974,045
Average	105,86		352,8	83,134	46,69	69,608	807,706
Std. Dev	59,30		48,55	5,009	1,704	8,351	118,2
13	7	Target 2	287,9	86,048	35,74	65,608	744,842
12	3		320,8	87,625	36,10	64,299	675,187
21	7		267,7	91,677	35,14	60,850	787,598
24	6		320,0	86,451	37,36	56,180	700,500
19	14		252,6	96,586	35,37	50,679	840,143
17	3		349,9	85,703	35,27	61,091	604,801
15	5		324,9	86,434	35,55	65,069	656,510
Average	6,43		303,4	88,646	35,79	60,539	715,654
Std. Dev	3,74		34,88	4,050	0,763	5,431	80,81

The agent performed relatively well for the first target. It found valid solutions on an average of approximately 106 episodes. The average FoM value was substantially above the set specification, which means that the algorithm naturally finds high quality solutions. For target 2, the agent performed exceptionally quickly, fitting all the constraints in a short amount of time. It balanced the low current and DC gain very well and still is able to find configurations that do not drop considerably in terms of GBW. The agent was also tested in this environment with a different architecture, model 2. The results are presented below:

Comparing the results to those obtained by model 1, there are slight differences, however, they are not impactful. For Target 1, model 2 finds a solution slightly faster than the first model. It also fulfills every constraint and the average FoM value is around 800. As for Target 2, the second model is also able to find a valid solution extremely quickly which fulfills every specified constraint. The average FoM value is higher than that of model 2. Much like VCOTA, there are no major differences in terms of performance between the two models. Results are also telling that the choice of seeds greatly affects the results.

To check the agent's progress over different episodes and visualize the learning of the agent during the sizing process, the same graphs were made for the FCA topology

Table 4.10 – Model 2, Target 1 - Performance results

Seed	# Episodes solved		IDD (µA)	DC Gain (dB)	GBW (MHz)	PM (º)	FoM (MHz×pF/mA)
12	209	Target 1	431,2	72,928	4,628E+07	75,011	643,970
72	129		383,4	82,489	4,690E+07	72,580	733,959
44	18		345,4	88,122	4,513E+07	53,354	783,961
56	32		422,4	78,134	4,944E+07	66,768	702,273
80	12		311,1	93,443	4,579E+07	52,046	883,124
21	116		318,2	83,715	4,543E+07	69,814	856,631
30	68		291,7	91,483	4,631E+07	54,155	952,554
Average	83,43		357,6E-04	84,331	4,647E+07	63,390	793,782
Std. Dev	72,08		55,51	7,312	1,438	9,890	109,3
Seed	**# Episodes solved**		**IDD (µA)**	**DC Gain (dB)**	**GBW (MHz)**	**PM (º)**	**FoM (MHz×pF/mA)**
13	18	Target 2	316,4	85,758	35,10	52,772	665,613
12	10		282,8	90,602	36,00	61,380	763,791
21	13		275,2	92,707	37,40	60,186	815,407
24	3		305,5	87,779	36,58	73,860	718,429
19	4		272,6	91,958	35,02	54,469	770,800
17	9		382,8	90,720	37,09	63,649	581,348
15	4		270,1	90,124	36,29	65,311	806,146
Average	8,71		300,8	89,950	36,21	61,661	731,648
Std. Dev	5,53		40,22	2,414	0,9152	7,060	83,92

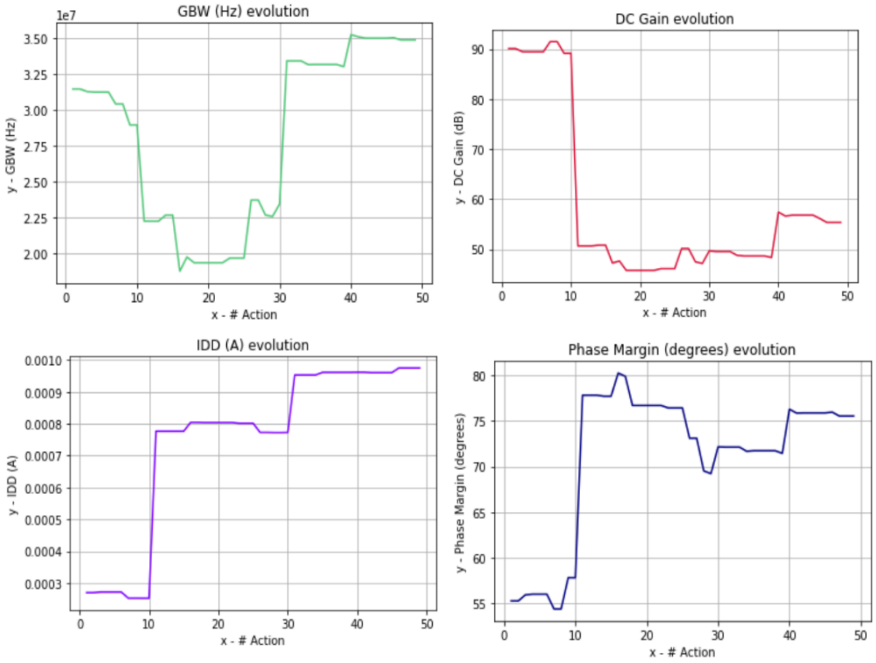

Figure 4.12 –FCA Model 1 – Early episode

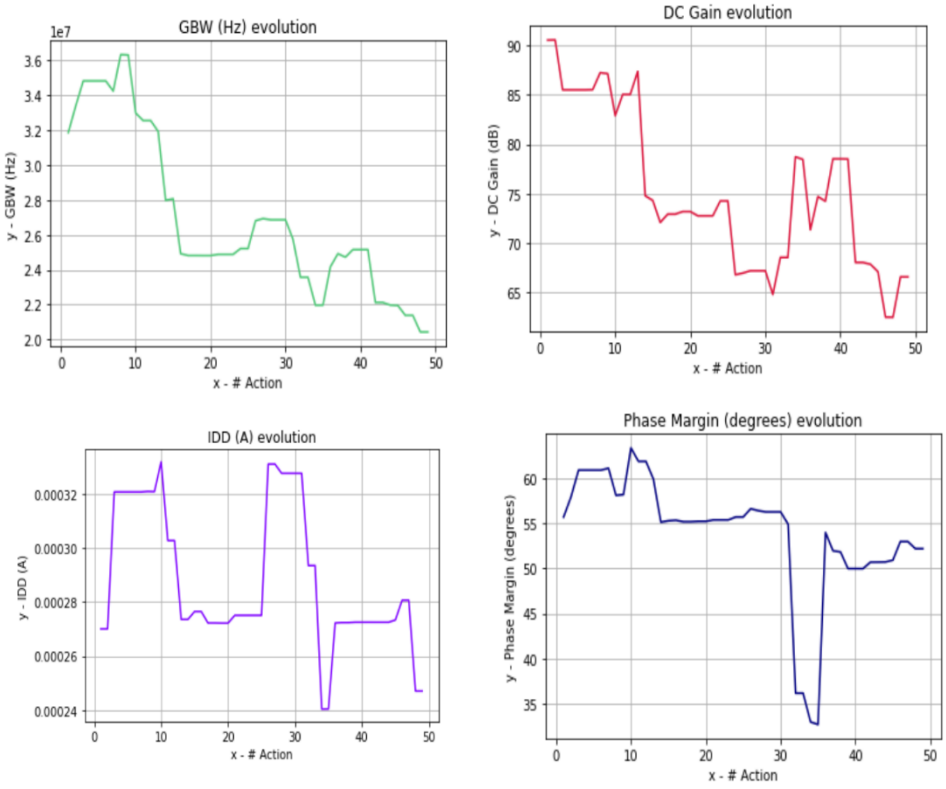

Figure 4.13 – FCA Model 1 – Intermediate Episode

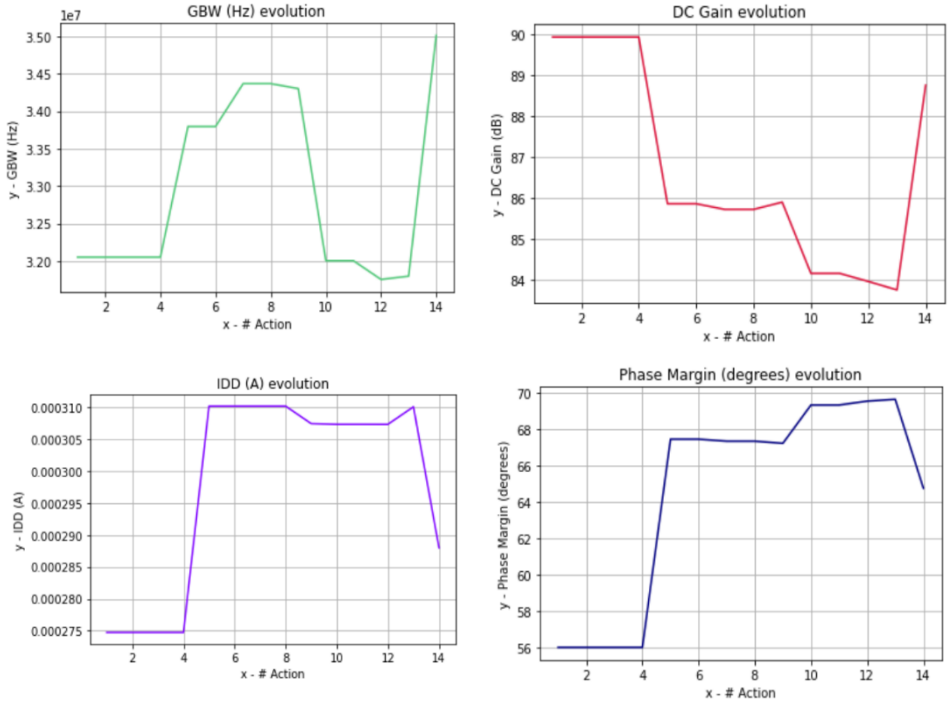

Figure 4.14 – FCA Model 1 – End Episode

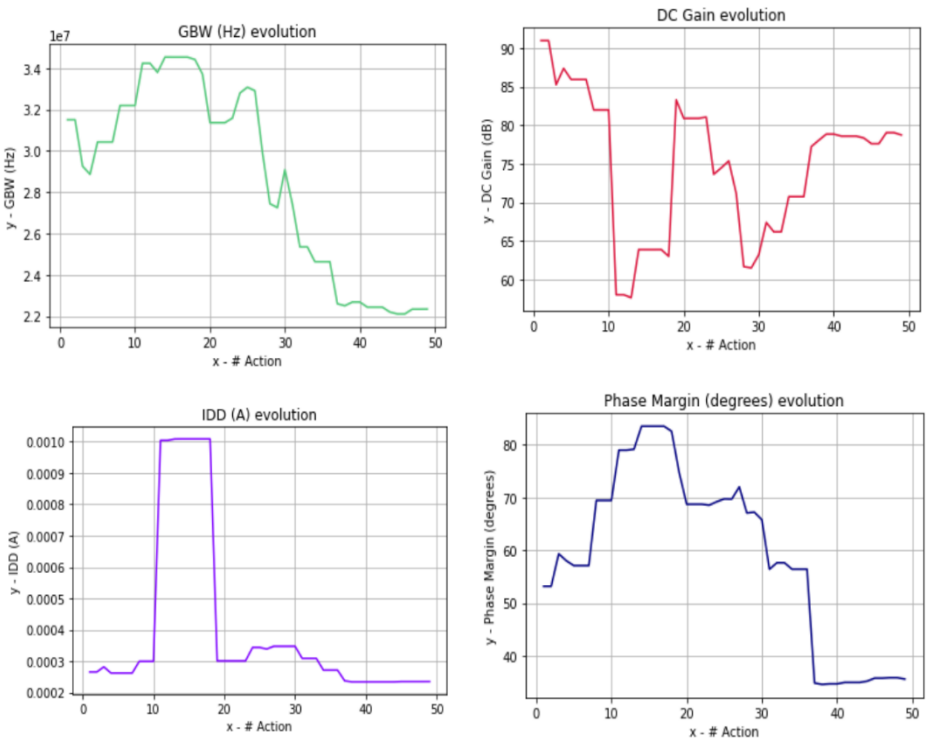

Figure 4.15 – FCA Model 2 – Early Episode

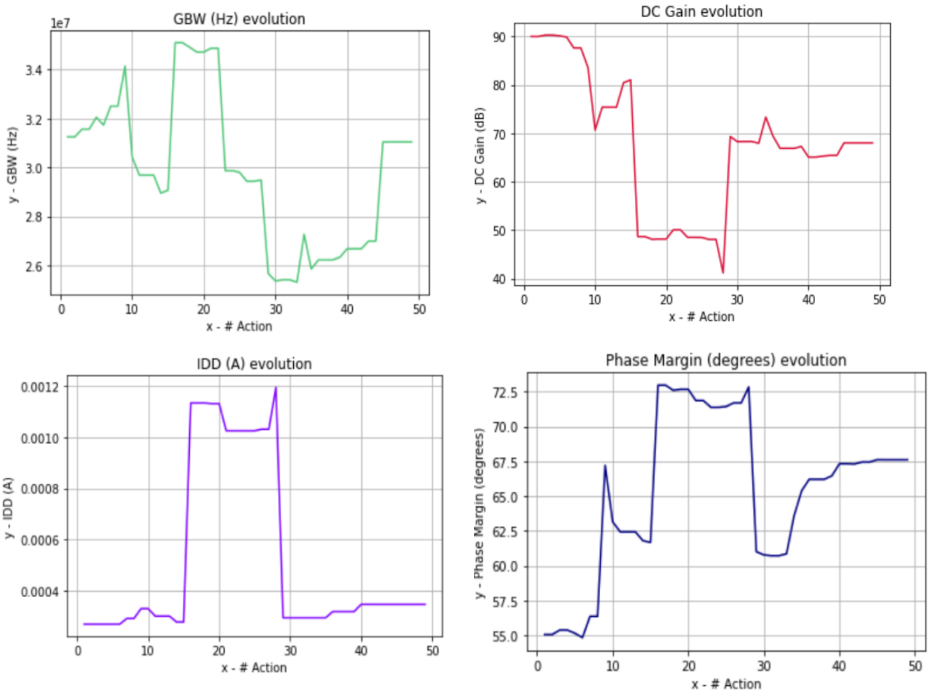

Figure 4.16 –FCA - Model 2 – Intermediate Episode

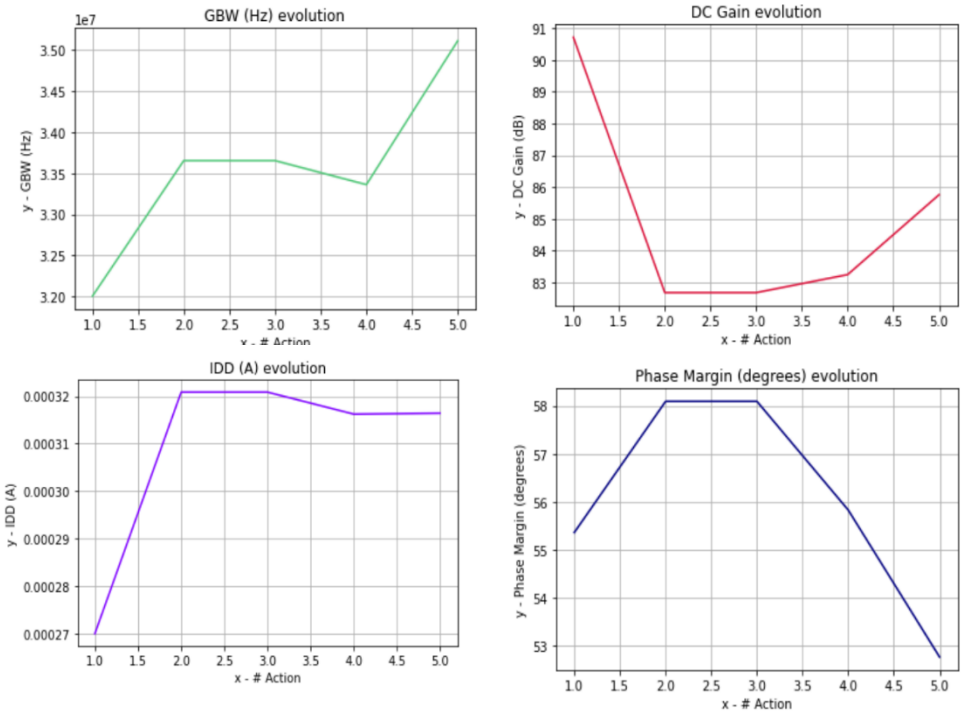

Figure 4.17 - FCA – Model 2 – End Episode

4.5 Agent Reutilization

In order to test whether the agent is capable of adapting to new circumstances, an experiment is performed. The main goal is assessing the agent's capabilities of adapting to new circumstances and if the performance improves if the agent acquires previous knowledge. The test consists of sizing a given topology for a different set of target specification after the agent's has successfully performed the sizing for an initial configuration of target specifications. For example, the agent can be performing the sizing process for Target 3 initially, and afterwards, without resetting the process, it performs the sizing process for Target 4. The process is performed for different pairings of target specifications to test how the agent adapts to different configurations. The results are presented in Table 4.11, Table 4.12 and Table 4.13 and are relative to the VCOTA topology:

Table 4.11 – VCOTA Model 1: Comparison between the number episodes taken to perform the sizing of Target 1 vs the number of episodes it takes to solve for Target 1 after the sizing is performed for Target 0

Seed	# Episodes Solved Target 1	# Episodes Solved for Target 1 after being sized for Target 0	
12	34	9	Target 0 → 1
22	12	39	
13	15	9	
16	77	47	
20	30	18	
Average	33,6	24,4	
Standard Deviation	26,0	17,6	

Table 4.12 – VCOTA Model 1: Comparison between the number episodes taken to perform the sizing of Target 1 vs the number of episodes it takes to solve for Target 4 after the sizing is performed for Target 3

Seed	# Episodes Solved Target 4	# Episodes Solved for Target 4 after being sized for Target 3	
24	41	33	Target 3 → 4
22	24	1	
13	103	40	
12	39	5	
Average	51,8	19,8	
Standard Deviation	35,0	19,6	

Table 4.13 – VCOTA Model 1: Comparison between the number episodes taken to perform the sizing of Target 1 vs the number of episodes it takes to solve for Target 2 after the sizing is performed for Target 1

Seed	# Episodes Solved Target 2	# Episodes Solved for Target 2 after being sized for Target 1	
24	32	1173	Target 1 → 2
12	34	28	
14	184	1014	
16	77	123	
Average	81,8	584,5	
Standard Deviation	71,3	592,6	

The three tables which are presented show three different scenarios. In the first case (Table 4.11), although slightly improving, the improvement in speed is not significant. Both configurations (Target 0 and Target 1) are balanced configurations, resulting in a multitude of beneficial actions which are not particularly identifiable. Therefore, although previous knowledge is useful, the improvement is not significant. In the second case (Table 4.12), there is a significant improvement. After performing the sizing for target 3 the agent quickly finds a solution for target 4. This is mainly due to the preference of actions that lower the I_{DD} value for both target specifications. Since the objective is the same, beneficial actions are not going to differ greatly and therefore, result in a significant improvement. The final case portraits a situation where there is a significant downfall in terms of results (Table 4.13). The two targets differ greatly in terms of preferential actions. Hence, the agent acquires knowledge beforehand that is not valuable knowledge for the next configuration and therefore needs to adapt during the following sizing process, resulting in an increase in time. Therefore, the agent is proven to be adaptable to different situations and is able to find a solution with prior knowledge although it is uncertain if the result will improve or not.

4.6 Conclusions

In this chapter, results for both topologies were presented. Each topology was subjected to agents which differed in architecture. The tested topologies were the VCOTA, a VC amplifier and a Folded Cascode. The agent was able to achieve valid solutions for the VCOTA circuit. The target specifications which were set for this circuit are enough to prove the agent's versatility. In a relatively small amount of time, the actor-critic method was able to find solutions for the problem at hand at increasingly difficult target specifications.

As for the folded cascode, the agent was also capable of finding several valid solutions in a short period of time. The actor-critic method adapted well to the target specifications that were set and quickly found configurations which successfully fit those constraints. In general, the agent performed very well for both topologies, proving that it can be generalized to different circuit topologies. This adaptation is very important in engineering, as it is vital for a method to perform under different circumstances.

5 Conclusions

This chapter presents the conclusions of all the work performed for this dissertation, and the future directions for the continuous development of Actor-Critic applied to analog IC sizing.

5.1 Work Conclusions

This work presents a reinforcement learning approach – Advantage Actor-Critic - that successfully contributed to the analog IC sizing for two amplifiers: VCOTA and FCA, given their intended target performances. There has not been much research on the topic at hand. Reinforcement Learning is still being tested on analog IC sizing. The available information is not widely spread as there is little research on the subject.

The Actor-Critic model proved to be very flexible, fast, and capable of satisfying complex constraints with relative ease. This technique can prove itself extremely useful in the future as it possesses a tremendous potential. This work shows that the actor-critic approach can learn design patterns and generate circuit sizing that are correct for specification trade-offs.

The limitless potential of the actor-critic algorithm shows considerable promises to future applications and could prove itself extremely useful in future research. This work showed that it can adapt to different analog IC environments, namely VCOTA and FCA, proving it generalizes quite well to other more complex environments of the same type, a feature which is extremely important in engineering.

On a final note, the proposed goals for this work were achieved, the actor-critic was able to successfully size the components of the two circuits, fulfilling all specified constraints.

5.2 Future Work

This work only scratches the surface of the impact the actor-critic method may have in Analog circuit sizing. There are still several opportunities where reinforcement learning might improve analog EDA. One great opportunity that might arise from this work is testing how the actor-critic method performs in more complex topologies. Assessing how it might behave in extremely complex circuits and experimenting more with it is promising. Additionally, assessing how other variants of the actor-critic algorithm behave and what results they might achieve is also a prospect that can be worth following.

Although the learning process is fast and efficient, it is important to test if the knowledge the agent stored during the sizing process can prove valuable and improve the speed as well as the transfer the knowledge to same circuit with different target specification or a different circuit. Once the agent has learned which actions improve the circuit's performance, the sizing process in theory should be faster and more efficient. Despite not being tested and implemented in this work, this is is a test which offers high potential and could further improve the agent's reliability and expand its versatility. For future reference, this is something that should be pursued as to improve the agent's capabilities and assess

how it would perform in different circumstances, such as adapting to new specifications or a new circuit after performing the sizing for a given set of specifications.

Other than testing how previous knowledge can influence future performances in different circumstances, one possibility that offers great potential is integrating Asynchronous Advantage Actor-Critic (A3C). A3C is an asynchronous version of A2C. Its main characteristic is allowing various agents to perform the sizing process separately and at the same time. This characteristic is advantageous since it enables the sizing of different topologies with different target specifications at the same time and using the same base model for the agent. Therefore, using A3C in the agent's development can be a step forward in the improvement of this work for future reference.

References

[1] The McClean Report - A complete analysis and forecast of the semicondutor industry. (2022). IC Insights. https://www.icinsights.com/data/reports/6/1/brochure.pdf?parm=1666909705.

[2] Lourenc¸o, Nuno, Ricardo Martins, and Nuno Horta. Automatic analog IC sizing and optimization constrained with PVT corners and layout effects. Cham: Springer International Publishing, 2017.

[3] Martins, Ricardo, et al." AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout generation." 2015 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD). IEEE, 2015.

[4] R. F. Badaoui and R. Vemuri," Analog VLSI circuit-level synthesis using multi-placement structures," 2005 IEEE International Symposium on Circuits and Systems, 2005, pp. 5978-5981 Vol. 6, doi: 10.1109/ISCAS.2005.1466001.

[5] K. Deb, A. Pratap, S. Agarwal, T. MeyarivanA fast and elitist multiobjective genetic algorithm: NSGA-II

[6] K. Deb, H. BeyerSelf-adaptive genetic algorithms with simulated binary crossover

[7] Evol. Comput., 9 (2) (2001), pp. 197-221, 10.1162/106365601750190406

[8] J. Domingues, A. Gusmão, N. Horta, N. Lourenço and R. Martins, "Accelerating Voltage-Controlled Oscillator Sizing Optimizations with ANN-based Convergence Classifiers and Frequency Guess Predictors," 2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2022, pp. 1-4, doi: 10.1109/SMACD55068.2022.9816265.

[9] IEEE Trans. Evol. Comput., 6 (2) (2002), pp. 182-197, 10.1109/4235.996017

[10] Zhao, Zhenxin, and Lihong Zhang." Deep Reinforcement Learning for Analog Circuit Sizing." 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2020.

[11] Degrauwe, Marc GR, et al. "IDAC: An interactive design tool for analog CMOS circuits." IEEE Journal of solid-state circuits 22.6 (1987): 1106-1116.

[12] Eldo Platform. (n.d.-b). Siemens Digital Industries Software. Retrieved October 31, 2022, from https://eda.sw.siemens.com/en-US/ic/eldo/

[13] Spectre Simulation Platform. (n.d.). Cadence. Retrieved October 31, 2022, from https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-simulation-platform.html

[14] Vogt, H. (n.d.). Ngspice, the open source Spice circuit simulator - Intro. Retrieved October 31, 2022, from https://ngspice.sourceforge.io/index.html

[15] 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, doi: 10.1109/43.31523.

[16] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for analog/rf circuit synthesis," in DAC, 18.

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

[18] Li, Yaguang, et al.” A Circuit Attention Network-Based Actor-Critic Learning Approach to Robust Analog Transistor Sizing.” 2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD). IEEE, 2021.

[19] Settaluri, Keertana, et al.” AutoCkt: deep reinforcement learning of analog circuit designs.” 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2020.

[20] Budak, Ahmet F., et al.” DNN-Opt: An RL Inspired Optimization for Analog Circuit Sizing using Deep Neural Networks.” 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE, 2021.

[21] Yang, Kai-En, et al.” Trust-Region Method with Deep Reinforcement Learning in Analog Design Space Exploration.” 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE, 2021.

[22] Mnih, Volodymyr, et al.” Asynchronous methods for deep reinforcement learning.” International conference on machine learning. PMLR, 2016.

[23] Schulman, John, et al.” Proximal policy optimization algorithms.” arXiv preprint arXiv:1707.06347 (2017).

[24] Schulman, John, et al.” Trust region policy optimization.” International conference on machine learn- ing. PMLR, 2015.

[25] A. Hill et al., “Stable baselines,” https://github.com/hill-a/stablebaselines, 2018

[26] H. Wang et al.,” GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning,” 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020, pp. 1-6, doi: 10.1109/DAC18072.2020.9218757.

[27] Mendes, Lu´ıs, et al.” In-Depth Design Space Exploration of 26.5-to-29.5-GHz 65-nm CMOS Low-Noise Amplifiers for Low-Footprint-and-Power 5G Communications Using One-and-Two-Step Design Optimization.” IEEE Access 9 (2021): 70353-70368.

[28] Brockman, Greg, et al.” Openai gym.” arXiv preprint arXiv:1606.01540 (2016).

[29] R. Gupta, B. M. Ballweber, and D. J. Allstot, “Design and optimization of CMOS RF power amplifiers,” IEEE J. Solid-State Circuits, vol. 36, no. 2, pp. 166–175, Feb. 2001.

[30] C. R. C. D. Ranter, G. van der Plas, M. S. J. Steyaert, G. G. E. Gielen, and W. M. C. Sansen, “CYCLONE: Automated design and layout of RF LC-oscillators,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 21, no. 10, pp. 1161–1170, Oct. 2002.

[31] G. Alpaydin, S. Balkir, and G. Dundar, “An evolutionary approach to automatic synthesis of high-performance analog integrated circuits,” IEEE Trans. Evol. Comput., vol. 7, no. 3, pp. 240–252, Jun. 2003.

[32] M. Chu and D. J. Allstot, “Elitist nondominated sorting genetic algorithm based RF IC optimizer,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 3, pp. 535–545, Mar. 2005.

[33] G. Tulunay and S. Balkir, “A synthesis tool for CMOS RF low-noise amplifiers,” IEEE Trans. Comput.- Aided Design Integr., vol. 27, no. 5, pp. 977–982, May 2008.

[34] Y. Xu, K.-L. Hsiung, X. Li, L. T. Pileggi, and S. P. Boyd, “Regular analog/RF integrated circuits design using optimization with recourse including ellipsoidal uncertainty,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 28, no. 5, pp. 623–637, May 2009.

[35] B. Liu, G. Gielen, and F. V. Fernandez, Automated Design of Analog and High-Frequency Circuits. Berlin, Germany: Springer-Verlag, 2014.

[36] R. Povoa, I. Bastos, N. Lourenc¸o, and N. Horta, “Automatic synthesis of RF front-end blocks using multi-objective evolutionary techniques,” Integr. VLSI J., vol. 52, pp. 243–252, Jan. 2016.

[37] R. Martins, N. Lourenco, N. Horta, J. Yin, P.-I. 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,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 1, pp. 69–82, Jan. 2019.

[38] F. Passos, R. Gonza'lez-Echevarr'ıa, E. Roca, R. Castro-Lopez, and F. V. Fernandez, "A two-step surrogate modeling strategy for single-objective and multi-objective optimization of radio frequency circuits," Soft Comput., vol. 23, pp. 4911–4925, Jul. 2018, doi: 10.1007/s00500-018-3150-9.

[39] G. G. E. Gielen, "Modeling and analysis techniques for system-level architectural design of telecom front-ends," IEEE Trans. Microw. Theory Techn., vol. 50, no. 1, pp. 360–368, Jan. 2002.

[40] D. R. de Llera Gonzalez, A. Rusu, and M. Ismail, "Receiver design for multi-standard wireless communications," in Radio Design in Nanometer Technologies, D. R. de Llera Gonza'lez and M. Ismail, Eds. New York, NY, USA: Springer, 2006.

[41] Wang, Jane & Kurth-Nelson, Zeb & Kumaran, Dharshan & Tirumala, Dhruva & Soyer, Hubert & Leibo, Joel & Hassabis, Demis & Botvinick, Matthew. (2018). Prefrontal cortex as a meta-reinforcement learning system. Nature Neuroscience. 21. 10.1038/s41593-018-0147-8.

[42] W. Sheng, A. Emira, and E. Sanchez-Sinencio, "CMOS RF receiver system design: A systematic approach," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 5, pp. 1023–1034, May 2006.

[43] S. Rodriguez, J. G. Atallah, A. Rusu, L.-R. Zheng, and M. Ismail, "ARCHER: An automated RF-IC Rx front-end circuit design tool," Anal. Integr. Circuits Signal Process., vol. 58, no. 3, pp. 255–270, Mar. 2009.

[44] Z. Pan, C. Qin, Z. Ye, and Y. Wang, "Automatic design for analog/RF front-end system in 802.11ac receiver," in Proc. 20th Asia South Pacific Design Autom. Conf., Chiba, Japan, Jan. 2015, pp. 454–459

[45] Zhang, Junzi, et al." Sample efficient reinforcement learning with REINFORCE." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 12. 2021.

[46] Shi, Wei, et al." RobustAnalog: Fast Variation-Aware Analog Circuit Design Via Multi-task RL." arXiv preprint arXiv:2207.06412 (2022).

[47] Chen, Po-Yan, et al." A Reinforcement Learning Agent for Obstacle-Avoiding Rectilinear Steiner Tree Construction." Proceedings of the 2022 International Symposium on Physical Design. 2022.

[48] K.-W. Lin et al., "A maze routing-based methodology with bounded exploration and path-assessed retracing for constrained multilayer obstacle-avoiding rectilinear Steiner tree construction," ACM TODAES, Vol. 23, Iss. 4, Article 45. May, 2018

[49] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multitask learning. arXiv preprint arXiv:2001.06782 (2020)

[50] H. -C. Jang, Y. -C. Huang and H. -A. Chiu," A Study on the Effectiveness of A2C and A3C Reinforcement Learning in Parking Space Search in Urban Areas Problem," 2020 International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 567-571, doi: 10.1109/ICTC49870.2020.9289269.

[51] Lourenc¸o, Nuno, 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). IEEE, 2018.

[52] Lourenc¸o, Nuno, et al." Using polynomial regression and artificial neural networks for reusable analog ic sizing." 2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD). IEEE, 2019.