

الجمهورية الجزائرية الديمقراطية  
الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث  
العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة سعيدة – د. الطاهر مولاي

University of Saida - Dr. MOULAY TAHAR

Faculty of Technology



A Dissertation Submitted to the Department of Telecommunications in Partial Fulfilment of  
the Requirements for Degree of Master of Network and Telecommunications

**Presented by:** Mr. Amrane Mohamed Belkebir

---

## **Python-Driven Deep Learning Approaches for Microstrip Filter Design**

---

Defended on September 23rd 2024 in front of the jury composed of:

Mr. Boudkhil Abdelhakim	Grade MCA	President
Mr. Chetoui Mohamed	Grade MCA	Supervisor
Mr. Damou Mehdi	Grade MCA	Examiner

**2023 / 2024**

## **Acknowledgements**

First of all, I would like to express my immense gratitude to my Almighty Allah, who has given me the courage, strength, health, and patience to accomplish this work. Alhamdulillah.

I also thank all the jury members for agreeing to evaluate this modest work. I extend my sincere thanks to my supervisor, Mr. CHETOUI & Doctorant Senasli Houda. I am deeply grateful to all the teachers who provided assistance during my university training years.

I would like to thank everyone who has helped me, whether directly or indirectly, even with just a simple piece of advice. Thank you all.

## *Dedication*

*To my dearest Mom and Dad,*

*Words cannot fully express my gratitude for your unwavering love, sacrifice, and belief in me. You have been my greatest inspiration, always encouraging me to aim higher and reach for my dreams. Every step I take and every success I achieve is a reflection of the foundation of strength, perseverance, and kindness you have instilled in me. Thank you for being my pillars of strength, guiding me with wisdom, and for always being there through every challenge.*

*To my friends Without Exception*

*Thank you for standing by my side through this incredible journey. You've been my source of laughter, support, and comfort, helping me navigate the ups and downs with courage and joy. Our shared memories and the bonds we've built will forever remain a cherished part of my life. You made this journey richer and more meaningful.*

*To my teachers, mentors, and all those who have touched my life,*

*Your guidance, wisdom, and encouragement have shaped me into who I am today. Thank you for pushing me beyond my limits and teaching me the values of hard work, curiosity, and perseverance.*

*Finally, to everyone who has offered their love and support, no matter how big or small this achievement is not mine alone but a testament to the collective belief you had in me*

## Contents

### Chapter I

1.1 Introduction:	11
1.2 Definition of microwave filtering	11
1.3 The Microstrip Technology:	12
1.4 Definition of the Filter and its Types:	12
1.4.1 High-Pass Filter:	12
1.4.2 Low-pass filter:	13
1.4.3 Bandpass Filter:	13
1.4.4 Band-stop filter:	14
1.5 Presentation of the filter:	14
1.5.1 Loss of insertions:	15
1.5.2 The bandwidth:	16
1.5.3 The attenuation band:	16
1.5.4 Cut-off frequency $f_c$ :	17
1.5.5 Central frequency $f_0$ :	17
1.5.6 Transfer function:	18
1.6. Ideal filter and real filter:	18
1.6.1 Template of a real filter:	19
1.6.2 Size of a filter's dimensions:	20
1.7 Approximation of filters :	20
1.7.1 Butterworth Approximation:	20
1.7.2 Chebyshev Approximation:	21
1.7.3 Generalized Chebyshev or Elliptic Approximation:	23
1.8 Low-pass filter prototype:	29
1.9 Transpositions Of frequency and impedance starting from the low-pass template:	32
1.10 Low-pass filter prototype:	35
1.11. Normalization and change of variable or frequency:	36
1.12. Impedance and Admittance Inverters:	36
1.13 Advantages and disadvantages of microstrip filters:	39
1.14 Microstrip Technology:	40
1.15. Effective dielectric constant( $\epsilon_{re}$ ) and characteristic impedance( $z_c$ ):	43

## Summary

---

1.16 Ondes de surface et modes d'ordre supérieur : .....	45
1.17 Discontinuities: .....	47
1.18. Advantages and disadvantages: .....	48
1.19 Conclusion .....	49
Chapter II	
2.INTRODUCTION: .....	51
2.2. Convolutional Neural Networks (CNNs): .....	51
2.3 ANN (Artificial Neural Networks): .....	58
2.4 Conclusion: .....	64
Chapter III	
3.1 Introduction: .....	66
3.2 Methodology: .....	66
3.3 Data preparation .....	67
3.4. Deep learning model development and architecture .....	73
3.5. Development environment .....	75
3.6. Model Training: .....	76
3.7. Results and discussion .....	84
3.8. Conclusion: .....	88
General Conclusion .....	90
Bibliography .....	93

## Figures Liste

### Chapter I

Figure 1.1 High-pass filter signal.....	12
Figure 1.2 signal of a low-pass filter .....	13
Figure 1.3 Band-pass filter signal	13
Figure 1.4 Band-pass filter signal.....	14
Figure 1.5 general structure of the filter.....	14
Figure 1.6: Observation of insertion losses based on the electrical response in transmission of a resonator. ....	16
Figure 1-7: Response of a band-pass filter with its main characteristics.	17
Figure 1.8: Response of a real band-pass filter.....	19
Figure 1.9: Filter templates.....	19
Figure 1-10 Frequency response of a Butterworth low-pass filter (Butterworth filter with $N=1, 2, 3, 4, 5, 6$ ) .....	21
Figure 1.11 Response of the Chebyshev function for different orders $n$ .....	22
Figure 1.12 the insertion losses of the Chebyshev prototype filter for $n=3, 5, 7$ , and $9$ .....	22
Figure 1.13 the parameters of an elliptical filter response .....	23
Figure 1.14 shows a low-pass elliptical filter.....	24
Figure 1.15 shows the insertion losses of a prototype elliptical filter for $n=3, 5$ , and $7$ ( $k=0.9, \epsilon=0.5$ )......	25
The prototype of the low-pass filter:.....	25
Figure 1.16 the prototype of the low-pass filter .....	25
Figure 1.17. The prototype low-pass filter of the elliptic function with (a) parallel resonators connected in series (b) series resonators connected in parallel. ....	28
Figure 1.18: Impedance Transformation.....	29
Figure 1.19: Low-pass to high-pass transformation.....	29
Figure I.20 Low-pass to band-pass frequency transformation.....	31
Figure 1.21 Impedance inverters (a) and admittance inverters (b).....	34

Figure 1.22 Generalized bandpass filters. (a) Impedance inverters K (b), admittance inverters J.....	34
Figure 1.25 Equivalent electrical diagram for a lossless narrow-band band-pass filter, showing all possible couplings. ....	35
Figure 1.26: structure of a microstrip bandpass filter with coupled lines.....	36
Figure 1.28 Modified low-pass filter prototypes to include admittance inverters. ....	37
Figure 1.29 Band-pass filter using admittance inverters .....	38
Figure 1.30 immittance inverters with localized elements.....	39
Figure 1.31: (a) Configuration of a line in microstrip technology (b) map of the fields. ....	40
Figure 1.32: The construction of a microstrip line. ....	41
Figure 1.33: Quasi-TEM Terrain Configurations .....	42

## Chapter II

Figure 2.1: Image arranged spatially: width, height, and depth.....	51
Figure 2.2: How CNN works.....	52
Figure 2.3: Principe of kernels .....	53
Figure 2.4: zero padding.....	54
Figure 2.5: mirror padding.....	55
Figure 2.6: max pooling.....	55
Figure 2.7: Relu function.....	56
Figure 2.8: A simple neural network .....	58
Figure 2.9: Activation functions.....	60
Figure 2.10: Operation of the Back propagation Algorithm [20].....	63
Figure 2.11: Activation functions.....	67
Figure 2.12: Operation of the Back propagation Algorithm [20].....	68
Figure 2.13: The Relationship between the Convolutional Layer and the Fully Connected Layer (ANN) [21].....	70

## Chapter III

Figure III.1: Microstrip Design simulation .....	69
Figure III.2: Microstrip coupled line structure.....	69
Figure III.3: Layout of the designed edge-coupled microstrip filter on .....	70
Substrate with a relative dielectric constant of 10.5. ....	70
FigureIII.4: Plot of S11&S21 as function with frequency.....	70
Figure III.5: Data before preprocessing .....	72
Figure III.6: Data after preprocessing.....	72
Figure III.7: Proposed CNN Topology for our study.....	73
Figure III.8: Neural Network structure of a single neuron in our Study.....	74
Figure III.9: schematic of the proposed deep neural network parametric model's overall development process.....	78
Figure III.10: model's learning progress over the epochs.....	84
Figure III.11: Plot of S11 real part & frequency .....	85
Figure III.12: Plot of S11 imaginary part & frequency.....	85
Figure III.13: Plot of S21 real part & frequency .....	86
Figure III.14: Plot of S21 imaginary part & frequency.....	87
Figure III.15: Magnitude & phase of S11, S21 plotted as function of frequency .....	87
Figure III.16: HFSS & CNN model plotted as function of frequency of S11, S21. .....	88



## Abbreviation List

---

### List Of Abbreviation

<b>AI</b>	Artificial Intelligence
<b>ANN</b>	Artificial Neural Networks
<b>AM</b>	Amplitude Modulation
<b>BW</b>	Bandwidth
<b>CAD</b>	Computer-Aided Design
<b>CNN</b>	Convolutional Neural Network
<b>DL</b>	Deep Learning
<b>DNN</b>	Deep Neural Network
<b>E-plane</b>	Electric Plane
<b>EMR</b>	Electromagnetic Radiation
<b>FFNN</b>	Feedforward Neural Network
<b>FC</b>	Fully Connected layer
<b>FM</b>	Frequency Modulation
<b>FNBW</b>	Fractional Null Beamwidth
<b>FNN</b>	Feedforward Neural Network
<b>GD</b>	Gradient Descent
<b>GRU</b>	Gated Recurrent Unit
<b>HFSS</b>	High-Frequency Structure Simulator
<b>HPBW</b>	Half Power Beamwidth
<b>IEC</b>	International Electrotechnical Commission
<b>IMUX</b>	Input Multiplexers
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>LSTM</b>	Long Short-Term Memory
<b>MMIC</b>	Monolithic Microwave Integrated Circuit
<b>ML</b>	Machine Learning
<b>MLP</b>	Multilayer Perceptron
<b>MSE</b>	Mean Square Error
<b>NLP</b>	Natural Language Processing

## Abbreviation List

---

<b>OMUX</b>	Output Multiplexers
<b>PCB</b>	Printed Circuit Board
<b>PEC</b>	Perfect Electric Conductor
<b>ReLU</b>	Rectified Linear Unit
<b>RF-EM</b>	Radio Frequency Electromagnetic
<b>RGB</b>	Red, Green, Blue
<b>RMSP</b>	Root Mean Square Propagation
<b>RNN</b>	Recurrent Neural Network
<b>SVM</b>	support vector machines
<b>SGD</b>	Stochastic Gradient Descent
<b>TEM</b>	Transverse Electric Magnetic
<b>3D</b>	Three-Dimensional

### Abstract

Python-Driven Deep Learning Approaches for Waveguide Filter Design (Microstrip) is a study that focuses on using Python programming and deep learning techniques to design waveguide filters. Waveguide filters are critical components in many communication and signal processing systems. The approach involves leveraging deep learning algorithms to optimize the design and performance of these filters. Python serves as the primary programming language, providing a versatile and powerful platform for implementing deep learning models. Deep learning algorithms, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), are employed to analyze and optimize waveguide filter designs. This research aims to streamline the design process, improve the efficiency of waveguide filters, and potentially discover novel filter designs that may not be apparent through traditional methods. By using Python and deep Learning.

**Keywords:** Microstrip, Deep learning, Filters, Convolutional neural network, Python, Hfss, Design.

### ملخص

الأساليب المعتمدة على بايثون في التعلم العميق لتصميم فلاتر الموجات الموجهة (الميكروستريب) هي دراسة تركز على استخدام برمجة بايثون وتقنيات التعلم العميق لتصميم فلاتر الموجات الموجهة. تعتبر فلاتر الموجات الموجهة مكونات حيوية في العديد من أنظمة الاتصالات ومعالجة الإشارات. تتضمن هذه الطريقة الاستفادة من خوارزميات التعلم العميق لتحسين تصميم وأداء هذه الفلاتر. تعتبر بايثون اللغة الأساسية للبرمجة، حيث توفر منصة متعددة الاستخدامات وقوية لتنفيذ نماذج التعلم العميق. تُستخدم خوارزميات التعلم العميق، مثل الشبكات العصبية التلافيفية أو الشبكات العصبية المتكررة، لتحليل وتحسين تصاميم فلاتر الموجات الموجهة. تهدف هذه البحث إلى تبسيط عملية التصميم، وتحسين كفاءة فلاتر الموجات، واكتشاف تصاميم جديدة للفلاتر قد لا تكون واضحة من خلال الطرق التقليدية. باستخدام بايثون والتعلم العميق

### الكلمات المفتاحية :

Hfss، الميكروستريب، التعلم العميق، الفلاتر، الشبكة العصبية التلافيفية، بايثون، التصميم

### Résumé

"Approches de conception de filtres à guide d'ondes (microstrip) basées sur l'apprentissage profond piloté par Python" se concentre sur l'utilisation de la programmation Python et des techniques d'apprentissage profond pour concevoir des filtres à guide d'ondes. Les filtres à guide d'ondes sont des composants essentiels dans de nombreux systèmes de communication et de traitement du signal. L'approche consiste à exploiter des algorithmes d'apprentissage profond pour optimiser la conception et les performances de ces filtres. Python est utilisé comme principal langage de programmation, offrant une plateforme polyvalente et puissante pour l'implémentation de modèles d'apprentissage profond. Des algorithmes d'apprentissage profond, tels que les réseaux de neurones convolutionnels (CNN) ou les réseaux de neurones récurrents (RNN), sont employés pour analyser et optimiser les conceptions de filtres à guide d'ondes. Cette recherche vise à rationaliser le processus de conception, à améliorer l'efficacité des filtres à guide d'ondes et potentiellement à découvrir de nouvelles conceptions de filtres qui ne seraient pas apparentes par des méthodes traditionnelles. Cela est rendu possible grâce à l'utilisation de Python et de l'apprentissage profond.

**Mots-clés:** Microstrip, Apprentissage profond, Filtres, Réseau de neurones convolutionnel, Python, Hfss, Conception.

# INTRODUCTION

### General Introduction

Antenna and microstrip patch design are essential elements in today's wireless communication systems. These technologies are key to facilitating effective signal transmission and reception, ensuring dependable connectivity across numerous devices and networks. Recently, researchers have made considerable progress in improving antenna performance and microstrip circuits. Nevertheless, as the demand for higher data rates, broader bandwidth, and greater efficiency continues to grow, it is crucial to explore new, innovative methods to further advance these technologies.

Neural networks and deep learning have shown to be effective techniques for handling challenging problems in a variety of fields. These methods enable the creation of extremely accurate predictive models because of their capacity to automatically identify and extract patterns from big datasets. Deep learning has shown impressive promise in speech recognition, natural language processing, and computer vision in recent years. Their use in the design of microstrip patches and antennas, however, is still largely unexplored. By utilizing deep learning techniques to transform the design process and empower engineers to optimize antenna settings, improve radiation patterns, and boost overall performance, this thesis aims to close this gap.

This work employs a deep learning approach to microstrip filter construction and analysis, along with a thorough examination of microstrip antennas. To enable a methodical approach, the study will be organized into three chapters:

The first chapter provides a comprehensive overview of filter theory, focusing on various fundamental parameters, types of filters, and their applications. The chapter begins with a microwave filter, with a definition of filters and their types. It then delves into the Approximation of Filters, including butterworth, chebychev and elliptique approximation.. The chapter then focuses on the transposition of frequency and impedance of low- and high-bandpass filters. Finally, the chapter delves into microstrip line (coupled line) structure, width marches, and the advantages and disadvantages of each.

The second chapter provides an introduction to the deep learning . Various domains, such as computer vision, pattern recognition, and natural language processing, have witnessed the tremendous power of deep networks. Next, the chapter explores Convolutional Neural Networks (CNNs).are a class of deep learning neural networks, that are widely used for tasks like image recognition, natural language processing, and other applications where detecting spatial or temporal patterns in data is crucial. It discusses how CNN works (what's happening in input & hidden & output layers). The chapter also focuses on Artificial Neural Networks (ANNs) . It explains the basics of Artificial Neural Network architecture, including input and output layers, hidden layers, and activation functions. It discusses the concepts of forward and backward propagation, weight initialization, and gradient descent optimization algorithms used in training Neural Networks. Finally, the chapter specifically explores the relation between convolutional and Fully Connected Neural Network.

The third chapter provides an experimental analysis of applying Deep Learning techniques to microstrip filter design. It begins by introducing the use of Deep Learning in microstrip design and detailing the methodology employed in the study. This includes the data preparation process, followed by a discussion on the development and architecture of the Deep Learning model, which encompasses the selection of suitable neural network architectures. The chapter then explains the model training process, covering the steps taken to train the Deep Learning model using the prepared dataset, as well as the optimization algorithms and the training and validation procedures used. In the results and discussion section, the performance of the trained model in microstrip design is presented, with a focus on analyzing and interpreting the accuracy and efficiency of the results. The chapter concludes with a summary of the key findings from the experimental analysis.

# Chapter One



### **1.1 Introduction:**

Microwave filters play a crucial role in modern telecommunications systems. They allow numerous applications (audio, video, and telecommunications, and instrumentation, radars) to share and make the best use of the limited resource that is the spectrum, particularly by helping to reduce interference between systems. Filtering a signal is a delicate operation that involves controlling the signal within a given frequency band, selecting the useful components (pass bands) and isolating those that are undesirable (stop bands). The ever-increasing capabilities of telecommunications satellites make the optimal use of the frequency spectrum necessary. The payload of a satellite is marked by several microwave filtering systems appearing at key locations: at reception, at the input multiplexers (IMUX), and at the output. (OMUX). Depending on the position of the filter, its specifications and the technology used to create it can vary significantly.

### **1.2 Definition of microwave filtering:**

Filtering is the action used to eliminate a frequency or a band of frequencies, or conversely, to enhance a frequency or a band of frequencies.

In other words, it is the action of modifying the spectral components of an electrical signal. Filters are the most commonly used components in the field of telecommunications, with their main application being the frequency multiplexing of signals.

The design of microwave filters is complicated because the components used have distributed parameters. There is no completely general synthesis method; the frequency behavior of microwave circuit elements (transmission lines, cavities, etc.) is complex, making it impossible to develop a general and comprehensive synthesis method.

Many microwave filter design techniques have been developed despite the additional complications posed by microwave frequencies. Microwave filters can be used in conjunction with other elements or passive devices, as is the case in multiplexers or duplexers often used in telecommunications. Microwave filters are also used in active circuits such as amplifiers, oscillators, etc.

### 1.3 The Microstrip Technology:

Microstrip technology widely used for creating microwave circuits, the microstrip structure consists of a metallic conductor deposited on the upper face of a dielectric substrate and a ground plane on the lower face [11]. The fundamental mode of propagation of such a propagation medium is not TEM (Transverse Electric Magnetic) because the transverse section is not homogeneous. However, since the amplitudes of the longitudinal components of the electric and magnetic fields are sufficiently small to be neglected, we refer to it as the quasi-TEM mode. This then makes it possible to model the structure in the form of a transmission line with characteristic impedance  $Z_c$  "immersed" in an equivalent homogeneous medium characterized by an effective relative permittivity  $\epsilon_{\text{eff}}$ .

### 1.4 Definition of the Filter and its Types:

The filter is an electronic circuit composed of an inductance and a capacitive resistance, characterized by a transfer function, which performs a signal processing operation. It is based on the coupling between several resonant cells that ultimately form a certain template in terms of losses, transmission, and reflection. It attenuates certain components of a signal in one frequency band and allows others to pass through in another frequency band called the pass band. Filters are classified by type as high-pass, low-pass, band-pass, and band-stop. [6]

#### 1.4.1 High-Pass Filter:

A high-pass filter is a filter that allows high frequencies to pass through while attenuating low frequencies.

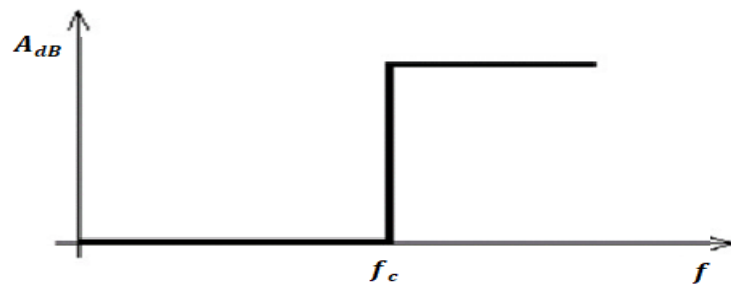


Figure I.1 High-pass filter signal

### 1.4.2 Low-pass filter:

A low-pass filter is a filter that allows low frequencies to pass through while attenuating high frequencies, meaning frequencies above the cutoff frequency. It could also be called a high-pass filter.

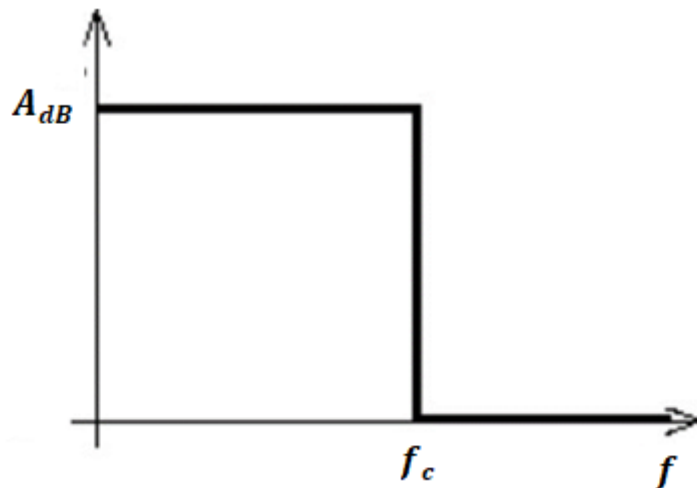


Figure I.2 signal of a low-pass filter

### 1.4.3 Bandpass Filter:

A band pass filter is a filter that only allows a band or range of frequencies to pass through, which is between a lower cutoff frequency and an upper cutoff frequency.

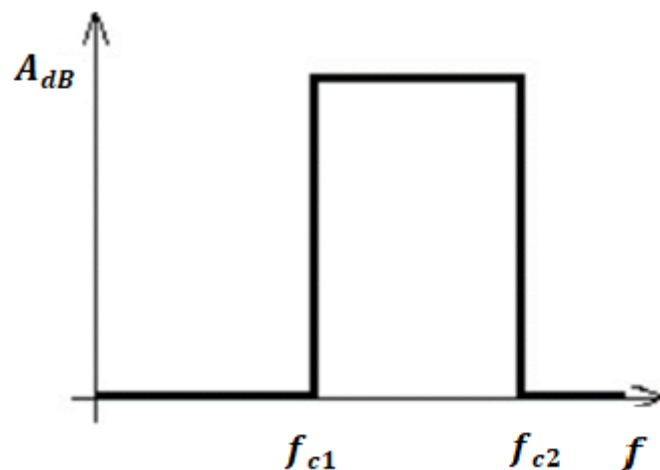


Figure I.3 Band-pass filter signal

## 1.4.4 Band-stop filter:

The last type of filter studied is the band-stop filter. This type of filter allows everything to pass through except for certain frequencies.

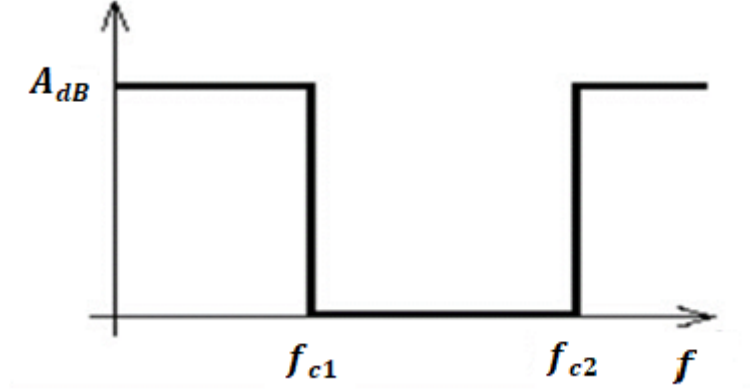


Figure I.4 Band-pass filter signal

## 1.5 Presentation of the filter:



Figure I.5 general structure of the filter

If port 2 of this device ends with a suitable load, we can relate the incident and output power as follows:

$$P_{out} = |S_{21}|^2 P_{inc} \quad (1.1)$$

We define this power transmission through a filter in terms of power transmission:

$$|S_{21}|^2 = \frac{P_{out}}{P_{inc}} \quad (1.2)$$

Microwave filters being generally passive, we observe that:

$$0 \leq \frac{P_{out}}{P_{inc}} \leq 1 \quad (1.3)$$

According to the law of conservation of energy:

$$P_{inc} = P_r + P_{abc} + P_{out} \quad (1.4)$$

With

$P_{inc}$ : Power incidente

$P_r$  : Reflected Power

$P_{abs}$ : power absorbed by the filter considered as loss

For a lossless microwave filter:

$$P_{inc} = P_r + P_{out} \quad (1.5)$$

We can write it with:

$$\frac{P_{inc}}{P_{inc}} = \frac{P_r + P_{out}}{P_{inc}} \quad (1.6)$$

$$1 = \frac{P_r}{P_{inc}} + \frac{P_{out}}{P_{inc}} \quad (1.7)$$

We define  $\frac{P_r}{P_{inc}}$  as the power reflection coefficient:

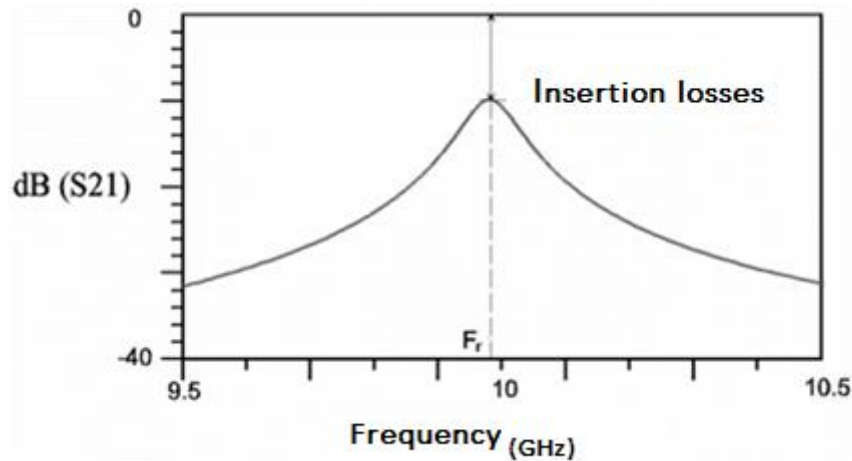
$$\Gamma = \frac{P_r}{P_{inc}} = |S_{11}|^2 \quad (1.8)$$

The filter outlet is equipped with a suitable load:

$$||S_{11}| + |S_{21}| = 1 \quad (1.9)$$

### 1.5.1 Loss of insertions:

Insertion losses are defined as the level of losses measured at resonance on the electrical response in transmission, which corresponds to the attenuation of the parameter  $|S_{21}|$  at the center frequency. (Figure2.5). Insertion losses are most often expressed in dB; however, there are times when they are reported or used in natural values.



**Figure I.6: Observation of insertion losses based on the electrical response in transmission of a resonator.**

### 1.5.2 The bandwidth:

It is equivalent to the frequency range at which the filter's insertion loss falls below a given value. For instance, the majority of the low-pass filter models made by Mini-Circuits are designed to have a maximum insertion loss value of 1 dB within the bandwidth.

### 1.5.3 The attenuation band:

Is the frequency range at which the insertion loss of the filter surpasses a given value? For example, the frequency range where the insertion loss is more than 20 dB and 40 dB in the band-stop filter characterizes the majority of Mini-Circuits low-pass filter models.

These two values were chosen at random; they could have easily been substituted for any other value. There are two objectives with the choice between 20 and 40 dB. The first is providing a simple way for the conceptual engineer to determine the frequency selectivity of the filter. Enabling speedy filter adequacy computations in particular scenarios is the second goal. These values were selected because in many systems, 20 dB or 40 dB reflect sufficient loss requirements. Based on frequency, actual loss values are provided in this manual. Les pertes de bande d'arrêt pour plusieurs mini-circuits filtres peuvent atteindre plus de 60 dB. [7]

### 1.5.4 Cut-off frequency $f_c$ :

It is the frequency where the insertion loss of the filter equals 3 dB. This is a very useful point for expressing the stop band and bandwidth boundaries. This offers a useful method for normalizing a filter's frequency response. For instance, the response obtained from dividing the frequency of a low-pass filter by  $ac$  would be "normalized" to  $ac$ . The design engineer can swiftly identify the filter required to satisfy their system's needs thanks to the standardized response. [6]

### 1.5.5 Central frequency $f_0$ :

It is the frequency around which band-pass filters are geometrically centered. For example, if  $f_1$  and  $f_2$  represent the 3 dB frequency points of a band-pass filter, then the central frequency  $f_0$  is calculated as follows:

When the bandwidth,  $f_2 - f_1$ , is a small percentage of the value of  $f_0$ , then  $f_0$ , which is equal to the geometric mean of  $f_2$  and  $f_1$ , will be approximately equal to their arithmetic mean in between.[10]

$$f_0 = \sqrt{f_1 \cdot f_2} \quad (1.10)$$

For a band-pass filter, the amplitude of the transfer function is maximum at the central frequency.

- Bandwidth  $\beta$ : It is the width of the bandwidth.
- Quality factor  $Q$ : It is the ratio of the central frequency to the bandwidth.

The quality factor is a measure of the bandwidth, regardless of the central frequency.

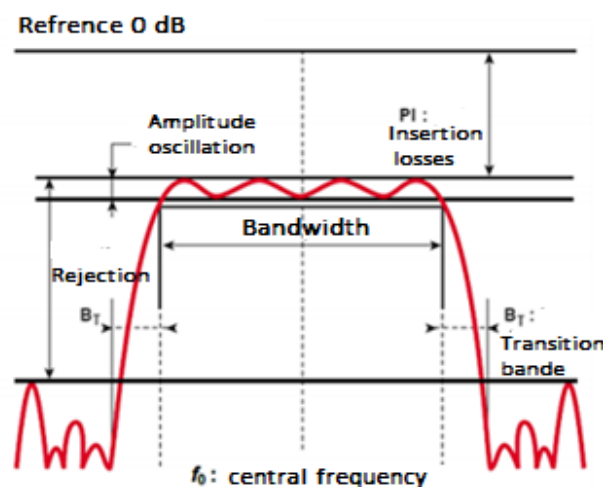


Figure I.7: Response of a band-pass filter with its main characteristics.

### 1.5.6 Transfer function:

The transfer function of a filter is a mathematical description of the response characteristics of a system, and mathematically it is the expression of  $S_{21}$ . The transfer function in squared amplitude for a lossless passive filter is defined by [8]:

$$|S_{21}(j\Omega)|^2 = \frac{1}{1 + \epsilon^2 F_n^2(\Omega)} \quad (1.11)$$

Where  $\epsilon$  is the wave constant,  $F_n$  represents the filtering or characteristic function, and  $\Omega$  is the frequency. The filter insertion loss response is calculated by:

$$L_A(\Omega) = 10 \log |S_{21}(j\Omega)|^2 \quad (1.12)$$

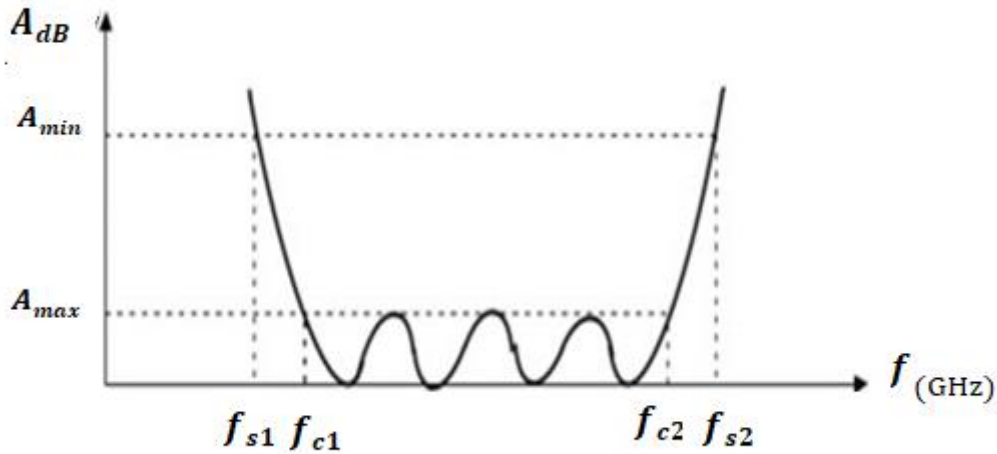
### 1.6 Ideal filter and real filter:

The creation of a filter requires knowledge of the frequency spectrum that makes up the useful signal; the ideal filter will transmit all components of the spectrum without distortion while completely eliminating unwanted signals. [3]

We cannot create filters that meet the ideal curves, but it is possible to bring them closer by acknowledging three imperfections:

- The attenuation in the pass band is not zero.
- The attenuation in the stop band represents an infinite value.
- The transition between the pass bands and the stop bands does not occur abruptly but rather gradually. [3]





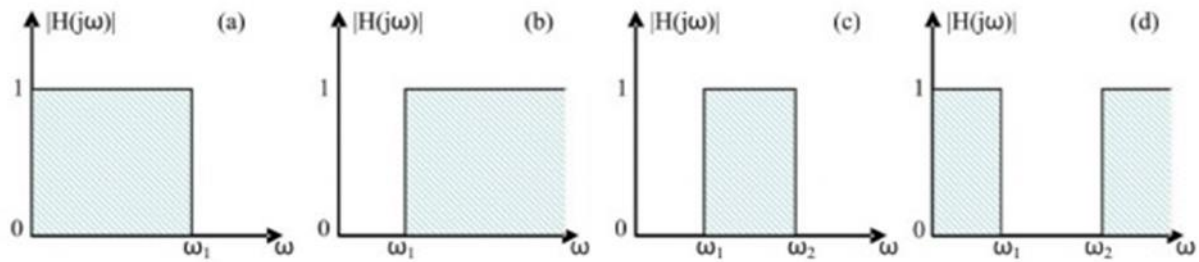
**Figure 1.8: Response of a real band-pass filter**

## 1.6.1 Template of a real filter:

The closer a filter is to the ideal filter, the steeper the transition bands become (low  $A_{max}$ , high  $A_{min}$ ) and the more components it requires.

So, the realization of a filter involves seeking a compromise between performance and the number of components. To achieve this goal, a template is defined within which the attenuation curve must be located. [3]

The different templates of the types of filters look like those in figure 2.9.



**Figure 2.9: Filter templates**

- a) Low pass
- b) High pass
- c) Band pass
- d) Band stop

### 1.6.2 Size of a filter's dimensions:

A low-pass (high-pass) filter template is fully defined by the knowledge of the quantities  $A_{max}$ ,  $A_{min}$ ,  $F_S$ ,  $F_C$ .

Regarding band-pass filters (band-stop), there are four frequencies ( $f_{s1}$ ,  $f_{s2}$ ,  $f_{c1}$ ,  $f_{c2}$ ) and two attenuations  $A_{max}$ ,  $A_{min}$  due to their symmetry around the central frequency.

These filters verify the relationship:

$$F_{C1}F_{C2} = F_{S1}F_{S2} = F_0^2 \quad (1.13)$$

$F_0$ : étant la fréquence centrale du filtre.

### 1.7 Approximation of filters:

By applying frequency changes to the low-pass prototype, three distinct filter types are produced: band pass, band stop, and high-pass. Because the intended transfer function and the rejection function both depend on the filter's order  $n$ , the theoretical synthesis of a filter starts with the synthesis of the related low-pass filter. By inserting various poles and zeroes, it is possible to design transfer functions with diverse features. We will thoroughly discuss the generalized or pseudo-elliptical Chebyshev, Butterworth, and Chebyshev functions utilized for both band-pass and low-pass filters here.

#### 1.7.1 Butterworth Approximation:

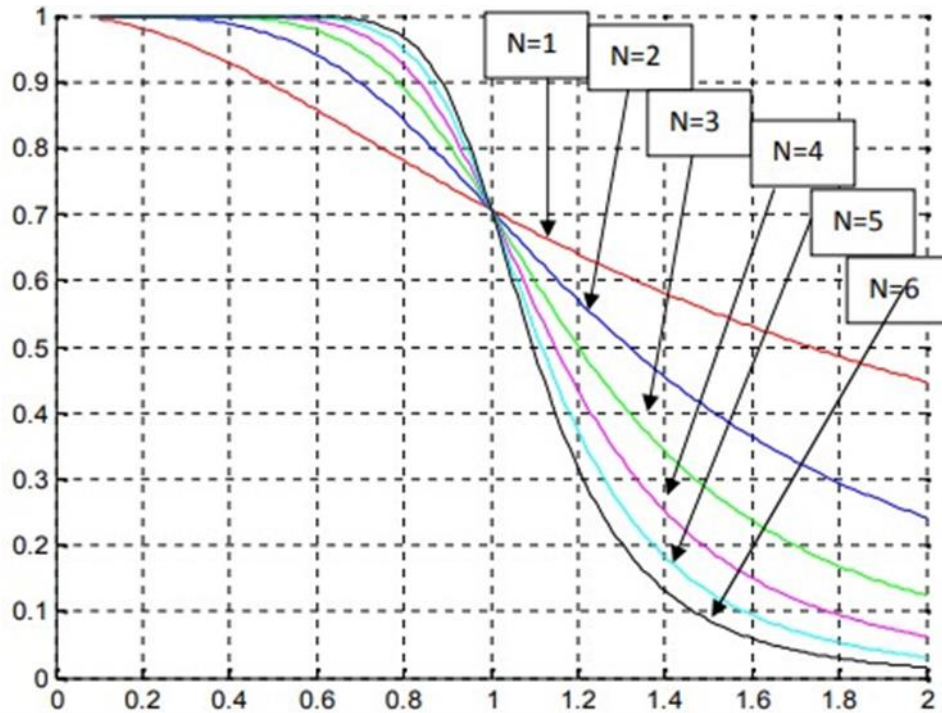
A Butterworth filter is a type of linear filter model, designed to have. A gain as constant as possible in its bandwidth.

For the Butterworth approximation, the transfer function of an  $n$ -the order filter is expressed as follows [8], [9]:

$$|H(j\omega)|^2 = \frac{1}{1 + (\omega/\omega_c)^{2n}} \quad (1.14)$$

$\omega_c$  Is the cutoff frequency,  $H_0$  is the attenuation constant. The parameters of the Butterworth filter are provided in Figure 1.3.

Figure 1.10 shows the insertion losses of the Butterworth filter for different filter orders.



**Figure 1-10 Frequency response of a Butterworth low-pass filter (Butterworth filter with N=1, 2, 3, 4, 5, 6)**

For various order n Figure 1.10 illustrates how we approach the ideal case of the filter as the filter's order increases.

### 1.7.2 Chebyshev Approximation:

The ability of Chebyshev filters to allow ripple in the pass band or stop band distinguishes them from other filter types. The ripple in the pass band is an approximate Type 1 Chebyshev filter. The following is the expression for the transfer function of a type 1 filter of order n [8] [9]:

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2 C_n^2(\omega/\omega_c)} \quad (1.15)$$

$$C_n(\omega) = \cos(n \cos^{-1}(\omega)) \quad (1.16)$$

The insertion losses of the Chebyshev filter for various filter orders are displayed in Figure 1.11:

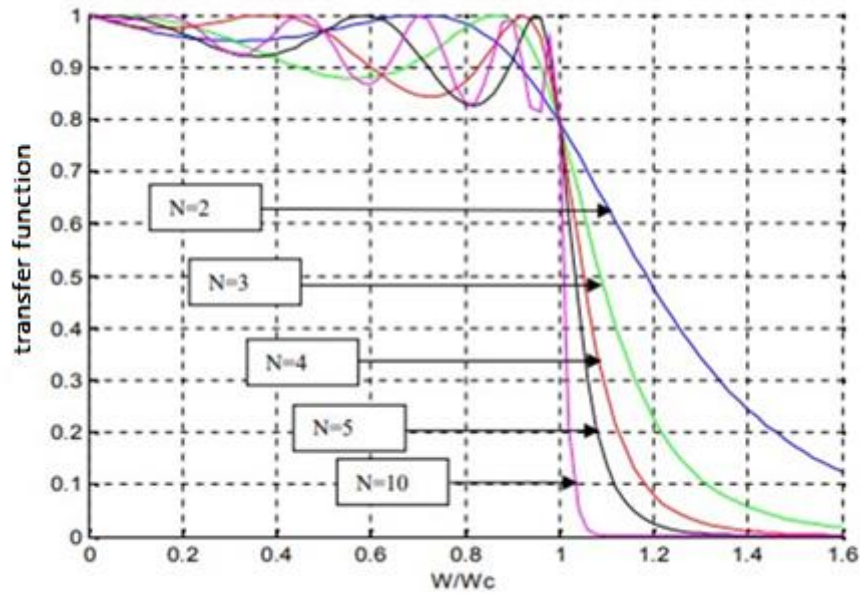


Figure 1.11 Response of the Chebyshev function for different orders n

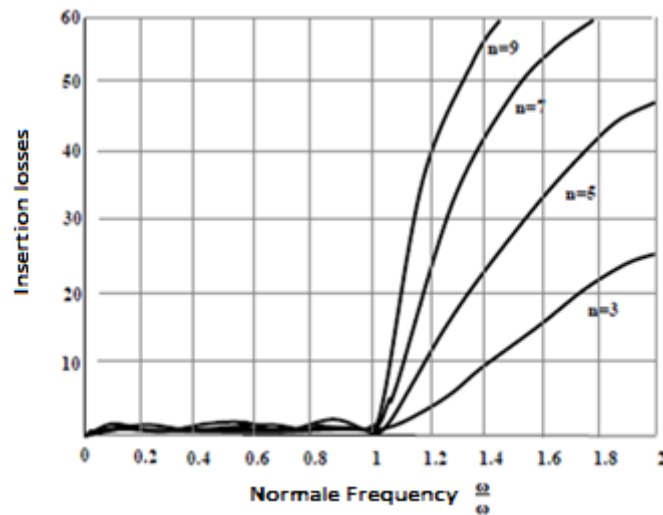


Figure 1.12 the insertion losses of the Chebyshev prototype filter for n=3, 5, 7, and 9. The amplitude response of a low-pass filter is specified by the transmission coefficient.

$$|S_{21}(j\omega)|^2 = \frac{1}{1 + \epsilon^2 C_n^2(\omega)} \quad (1.17)$$

$$\epsilon = \sqrt{10^{\frac{A_{Max}}{10}} - 1} \quad (1.18)$$

$\epsilon$  is a parameter that characterizes the ripple in the filter's bandwidth, and by replacing the value of the insertion loss  $A_{Max}$  expressed in (dB).

$$C_n(\omega) = \begin{cases} \cos(ncos^{-1} \omega), & |\omega| \leq 1 \\ \cosh(ncosh^{-1} \omega), & |\omega| \geq 1 \end{cases} \quad (1.19)$$

$C_n(\omega)$  Is the Chebyshev polynomial.

### 1.7.3 Generalized Chebyshev or Elliptic Approximation:

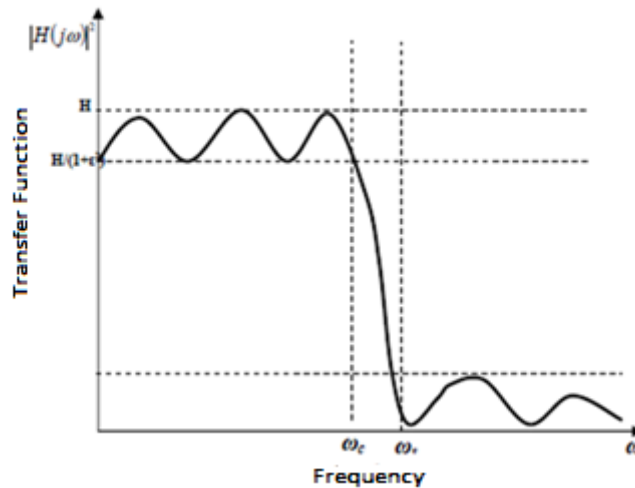
Generalized Chebyshev functions are more complex but allow for better rejections because they have, like the Chebyshev function, reflection zeros within the band, as well as transmission zeros outside the band at frequencies  $\omega_i$  .[6]

$$|S_{21}(j\omega)|^2 = \frac{1}{1 + \epsilon^2 F_n^2(\omega)} \quad (1.20)$$

Where  $F_n(\omega)$  is the filtering function and it is given by

$$F_n(\omega) = \begin{cases} M \frac{\prod_{i=1}^{\frac{n}{2}} \omega_i^2 - \omega^2}{\prod_{i=1}^{\frac{n}{2}} \frac{\omega_c^2}{\omega_i^2} - \omega^2} & n=pair \\ N \frac{\prod_{i=1}^{\frac{n-1}{2}} \omega_i^2 - \omega^2}{\prod_{i=1}^{\frac{n-1}{2}} \frac{\omega_c^2}{\omega_i^2} - \omega^2} & n \text{ impair} \geq 3 \end{cases} \quad (1.21)$$

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2 F_n^2(\omega/\omega_c)^{2n}} \quad (1.22)$$



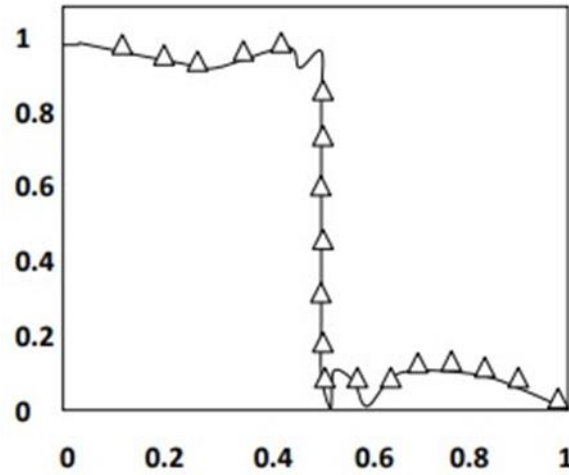
**Figure 1.13 the parameters of an elliptical filter response**

The elliptical approximation is distinguished from others by its characteristic of equal undulation both in the pass band and in the stop band [5], [6].

Moreover, it has transmission zeros in its electrical response that allow for a good level of filter selectivity with a limited order. It is defined by its attenuation function:

$$\alpha_a = 10 \log[1 + \varepsilon^2 \varphi_n^2(w)] \quad (1.23)$$

Where the function  $\varphi_n$  is an elliptic function of order  $n$  and  $\xi$  is a parameter that determines the ripple in the bandwidth  $w$  at the cutoff frequency.



**Figure 1.14 shows a low-pass elliptical filter.**

A closer look at the three filters' curves reveals that the Butterworth filter has a lower rejection, suggesting that a higher order is required for a particular implementation. Conversely, its sound is noticeably more constant throughout the entire frequency range. In cases where a sharp transition is needed, elliptical and Chebyshev filters are employed. In contrast to inverse Chebyshev filters, also known as type 2 filters, which display ripple in the stopband, Chebyshev filters display ripple in the passband. Even with their improved selectivity, elliptical filters ripple in both bands and are frequently more challenging to tune.

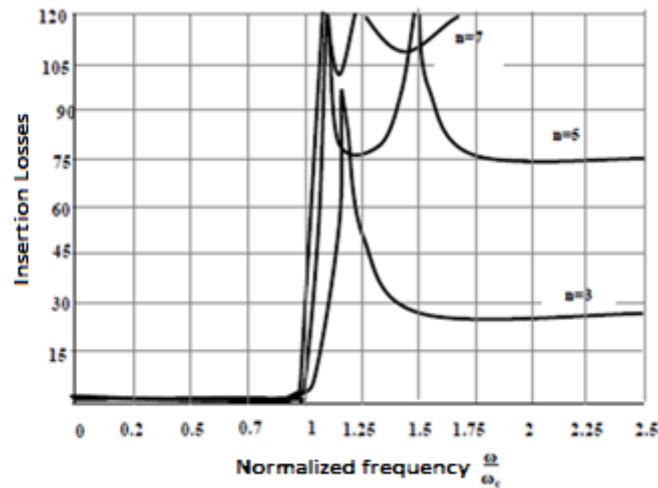


Figure 1.15 shows the insertion losses of a prototype elliptical filter for  $n=3, 5,$  and  $7$  ( $k=0.9, \epsilon=0.5$ ).

## 1.8 The prototype of the low-pass filter:

The transfer function calculation will be carried out on a prototype low-pass filter, which is a representative example of the filter to be produced, regardless of the type of filter (in the family of low-pass, high-pass, band-pass, and band-stop filters).

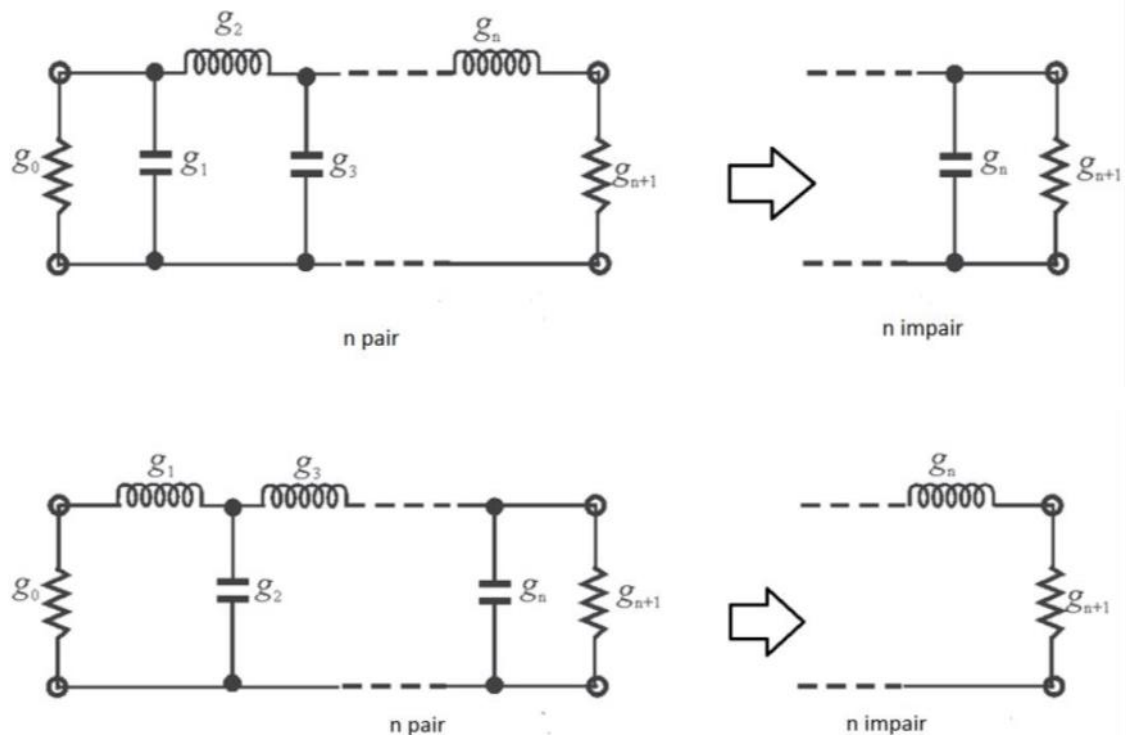


Figure 1.16 the prototype of the low-pass filter

### 1.8.1 Prototype low-pass filter of Butterworth:

✓ **The order of the filter**

For the design of a low-pass filter, the order  $n$  is determined by the following relationship:

$$n \geq \frac{\log(10^{0.1A_{Min}} - 1)}{2\log(\omega)} \quad (1.24)$$

Or:

$$\frac{\omega_a}{\omega_c} = 10^{0.1A_{Min}} - 1 \quad (1.25)$$

The values of the components of the prototype low-pass Butterworth filter with equal resistive terminations are determined using the following equations:

$$\begin{aligned} g_0 &= 1 \\ g_{n+1} &= 1 \end{aligned} \quad (1.26)$$

$$g_i = 2 \sin\left[\frac{(2i-1)\pi}{(2n)}\right]$$

for  $i = 1, 2, \dots, n$

Despite its simplicity, the Butterworth filter type h response is not widely used in practice due to insufficient selectivity.

### 1.8.2 Prototype of the Chebyshev low-pass filter:

✓ **The order of the filter:**

For the design of a low-pass filter, determine the order  $n$  using the following relationship:

$$N \geq \frac{\cosh^{-1}(\sqrt{(10^{0.1A_{Min}} - 1)/\epsilon})}{\cosh^{-1}(\omega_a/\omega_c)} \quad (1.27)$$

For an attenuation  $A_{min}$  at the frequency  $\omega_0$ . The  $g_i$  parameters are calculated as follows:

$$\begin{aligned} g_0 &= 1 \\ g_1 &= (2/\gamma) \sin(\pi/2n) \end{aligned}$$



$$g_n = \frac{1}{g_{i-1}} \frac{4 \sin\left[\frac{(2i-1)\pi}{2n}\right] \sin\left[\frac{(2i-3)\pi}{2n}\right]}{\gamma^2 + \sin^2\left[\frac{(i-1)\pi}{n}\right]} \quad (1.28)$$

$$g_{n+1} = 1 \quad \text{if } n \text{ is even} \quad (1.29)$$

$$g_{n+1} = \coth^2(\beta/4) \quad \text{if } n \text{ is odd} \quad (1.30)$$

$$\beta = \ln \left[ \coth \left( \frac{A_{Max}}{17.37} \right) \right] \quad (1.31)$$

$$\gamma = \sinh \left( \frac{\beta}{2n} \right) \quad (1.32)$$

### 1.8.3 Low-pass prototype of the elliptical filter:

Using stepped-impedance microstrip hairpin resonators and their associated circuit models, a small low-pass filter with an elliptical function is created. The analogous circuit concept is used to create prototype filters by using the element value tables that are readily available. The prototype filters' size is modified using an electromagnetic simulation to maximize filter performance. A very sharp cutoff frequency response and little insertion loss are offered by the filter that uses several cascaded hairpin resonators. Moreover, more attenuation poles are added to the filter to expand its bandwidth and rejection range. Good agreement is found in the evaluation of the filters based on simulation and experience. This kind of filter, along with others, may be designed and understood with the help of this straightforward equivalent circuit model.

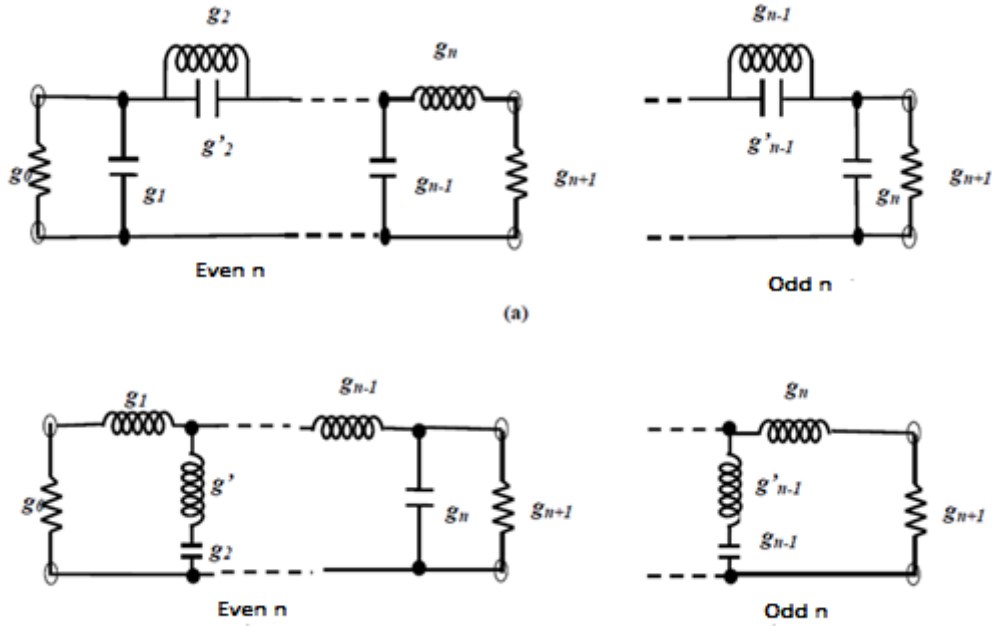


Figure 1.16. The prototype low-pass filter of the elliptic function with (a) parallel resonators connected in series (b) series resonators connected in parallel.

### 1.9 Transpositions Of frequency and impedance starting from the low-pass template:

Via frequency modifications, standardized low-pass . In fact, capacitance and inductance are the two components that make up a low-pass filter. As a result, the low-pass prototype is used to create the LC networks of filters (high-pass, band-pass, and band-stop) by simply changing the impedance and frequency.

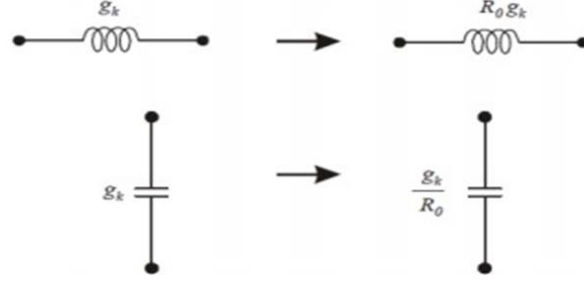
#### 1.9. 1 Transposition In impedance:

We have to denormalize the elements of the low-pass prototype since all of its elements ( $g_k$ ) are normalized with regard to frequency and impedance.

De-normalization of the impedance can be achieved easily by dividing the  $g_k$  representing parallel capacitors by  $R_0$  and multiplying the  $g_k$  representing series inductors by the load resistance  $R_0$  .

This denormalization of impedance is seen in figure 1.17 .

In order to create high-pass, band-pass, or band-stop filters from the low-pass prototype, we need to perform frequency transformation.



**Figure 1.17: Impedance Transformation**

## 1.9.2. Low-pass to high-pass transformation:

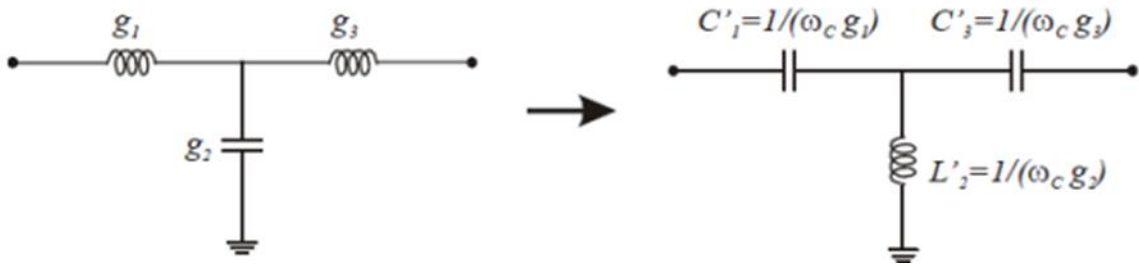
The frequency transformation from the low-pass plane ( $\omega$ ) to the high-pass plane ( $\omega'$ ) is defined by:

$$C'_K = \frac{1}{\omega_c L_K} = \frac{1}{\omega_c g_k} \quad (1.33)$$

$$L'_K = \frac{1}{\omega_c C_K} = \frac{1}{\omega_c g_k} \quad (1.34)$$

Under these conditions, the high-pass prototype can be easily deduced from that of the low-pass. To do this, each self must be replaced by a capacity and vice versa. Let the capacitances  $C_K$  and inductances  $L_K$  of the low-pass prototype be given, then the values (frequency-denormalized) for the capacitances  $C_K$ , and inductances  $L_K$ , of the high-pass filter are calculated by :

This transformation is illustrated in the figure.



**Figure 1.18: Low-pass to high-pass transformation**

## 1.9.3 Frequency transformation: low pass $\rightarrow$ band pass:

Let's suppose that a prototype low-pass response needs to be transformed into a band-pass response with a bandwidth of  $\omega_2 - \omega_1$ , where  $\omega_1$  and  $\omega_2$  indicate the cutoff frequencies of the band-pass filter. The template of a band-pass filter is defined by:

- At the central pulsation  $\omega_0$
- At the low cutoff pulsation  $\omega_1$ .

➤ At the high cutoff pulsation  $\omega_2$ .

Let:

The frequency transformation from the low-pass plane ( $\omega$ ) to the band-pass plane ( $\omega'$ ) is defined as:

$$\omega \rightarrow \frac{1}{\Delta} \left( \frac{\omega_0}{\omega'} - \frac{\omega'}{\omega_0} \right) \quad (1.35)$$

$$\omega_0 = \sqrt{\omega_1 \omega_2} \quad (1.36)$$

$$\Delta = \frac{\omega_2 - \omega_1}{\omega_0} \quad (1.37)$$

The obtaining of the band-pass filter from the low-pass prototype is done in two steps.

1- The inductances in series must be replaced by a series LC resonant circuit, with the values of the elements defined as follows.

$$C'_i = \frac{\Delta}{\omega_0 l_i} = \frac{\Delta}{\omega_0 g_i} \quad (1.38)$$




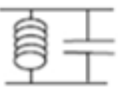
$$L'_k = \frac{l_i}{\Delta \omega_0} = \frac{g_i}{\Delta \omega_0} \quad (1.39)$$

2) The parallel capacities must be replaced by a parallel LC resonant circuit, with the values of the components defined as follows:

$$C'_i = \frac{c_i}{\Delta \omega_0} = \frac{g_i}{\Delta \omega_0} \quad (1.40)$$

$$L'_i = \frac{\Delta}{\omega_0 c_i} = \frac{\Delta}{\omega_0 g_i} \quad (1.41)$$

The frequency transformation from low-pass to band-pass involves the following modifications according to the equivalent circuit used:

Low-pass	Band-pass
 $L_n$ Low-pass	 $L_n$ Band-pass = $\frac{L_n \text{ Low-pass}}{W}$ $C_n$ Band-pass = $\frac{C_n \text{ Low-pass}}{W}$
 $C_n$ Low-pass	 $C_n$ Band-pass = $\frac{C_n \text{ Low-pass}}{W}$ $L_n$ Band-pass = $\frac{L_n \text{ Low-pass}}{C_n \text{ Low-pass}}$

$W = \omega_{high} - \omega_{Low}$   
(W:Real band of the band-pass)

Figure I.19 Low-pass to band-pass frequency transformation

#### 1.9.4. Band-stop filter transformation:

For the case of frequency transformation for the band-stop filter, we use the following transformation:

$$w \rightarrow -\Delta \left( \frac{w'}{w_0} - \frac{w_0}{w} \right)^{-1} \quad (1.42)$$

To obtain the band-stop filter, it is enough to do the opposite of the band-pass filter case, where you need to replace the inductances with the parallel LC circuit defined by:

$$C'_k = \frac{1}{\Delta L_K w_0} = \frac{1}{\Delta g_k w_0} \quad (1.43)$$

$$(1.44)$$

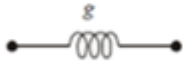
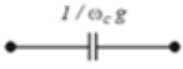
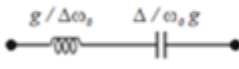
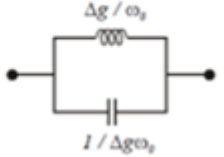
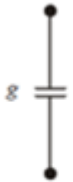

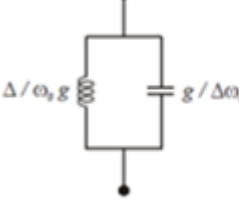
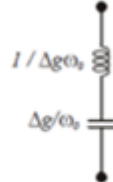
$$L_K = \frac{\Delta L_K}{w_0} = \frac{\Delta g_k}{w_0}$$

And replace the capacitors with the series LC circuit defined by

$$C'_K = \frac{\Delta C_K}{w_0} = \frac{\Delta g_k}{w_0} \quad (1.45)$$

$$L_K = \frac{1}{\Delta C_K w_0} = \frac{1}{\Delta g_k w_0} \quad (1.46)$$

All these frequency transformation steps can be summarized in Table 1.

Low-pass	High-pass	Band-pass	Band-stop
			
			

**Table 1.1 Frequency transformation from the low-pass prototype.**

## 1.10 Low-pass filter prototype:

The realization of any filter boils down to that of the low-pass filter called the "prototype."

To facilitate the synthesis, we introduce two simplifications:

- The normalization of frequency and impedance units
- Frequency transposition.

### 1.10.1 Normalization of units:

**1. Normalization of the frequency unit:** the normalized frequency is

$$f = \frac{F}{F_u}$$

For low-pass and high-pass filters,  $F_u$  will be the last passing frequency:  $F_u = F_c$

And for band-pass or band-stop filters, the central frequency is such that:

$$F_u = F_0 = \sqrt{F_{C1} F_{C2}} \quad (1.47)$$

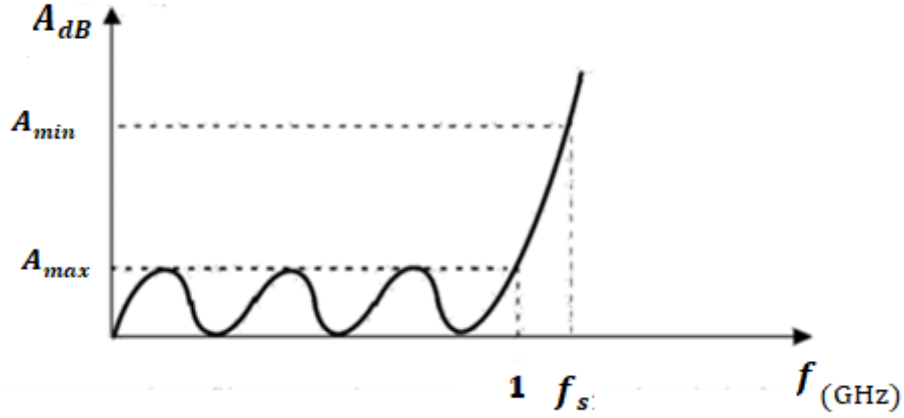


Figure 1.20: response of a low-pass filter expressed as a function of normalized frequency.

## 2. Normalisation Of Impedance Unit:

We take a particular value  $R_0$  as the unit of impedance, and normalization is generally done with respect to the load impedance  $R_L$ .

### 1.10.2 Values of the components of the low-pass filter:

The structure of the low-pass filter is shown in Figure 2.15.

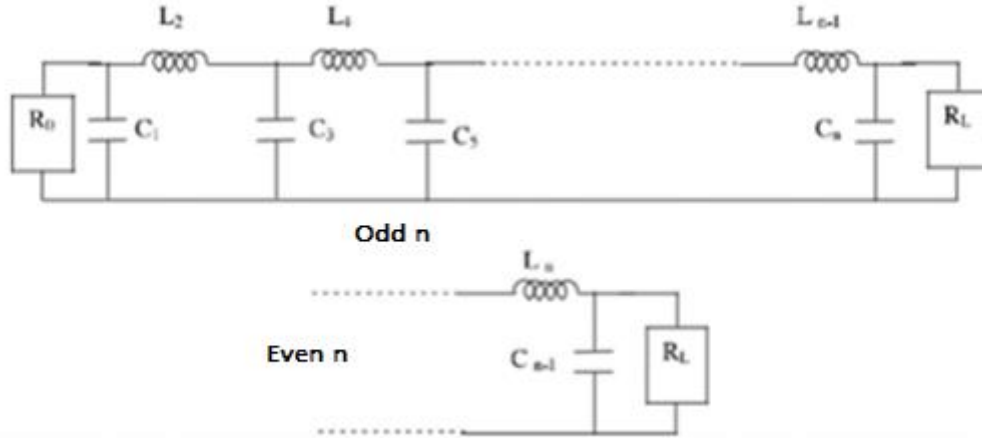


Figure 1.21: low-pass filter prototype.

$R_0$  is a parameter of the problem since it is the characteristic impedance of the line on which the filter is inserted, and we will benefit from the case where  $R_0$  equals  $R_L$ : this simplifies the calculation and allows for the same impedances at the input and output. The self-inductances  $L_k$  and the capacitances  $C_k$  of the low-pass prototype are expressed in terms of  $R_0$ ,  $w_c$  and the parameter  $g_k$  :

$$L_K = \frac{R_0}{w_0} g_k \quad \text{And} \quad C_K = \frac{1}{R_0 w_C} g_k \quad (1.48)$$

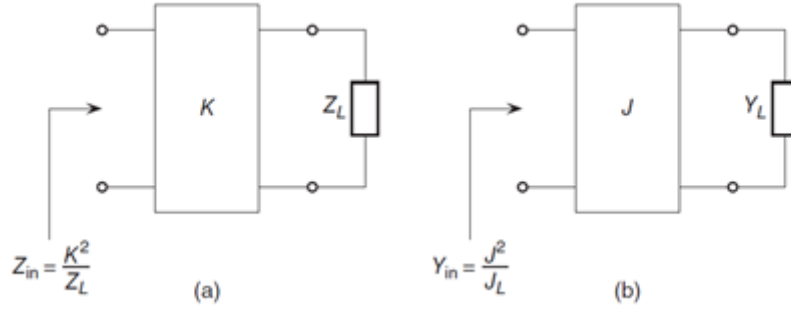
### 1.11. Normalization and change of variable or frequency:

From the synthesis of the normalized low-pass filter, characterized by the values of inductances  $L_n$  and capacitances  $C_n$ , one can obtain the synthesis of any filter for any central frequency  $\omega_0$ , any reference frequency band ( $\omega_{c2} - \omega_{c1}$ ), and suitable for an RL load.

We can therefore outline Table (1.1) in transformation diagrams allowing for the synthesis of filters whose amplitude and transition time curves are derived from the curves of the same nature of normalized low-pass filters through variable changes. [8]

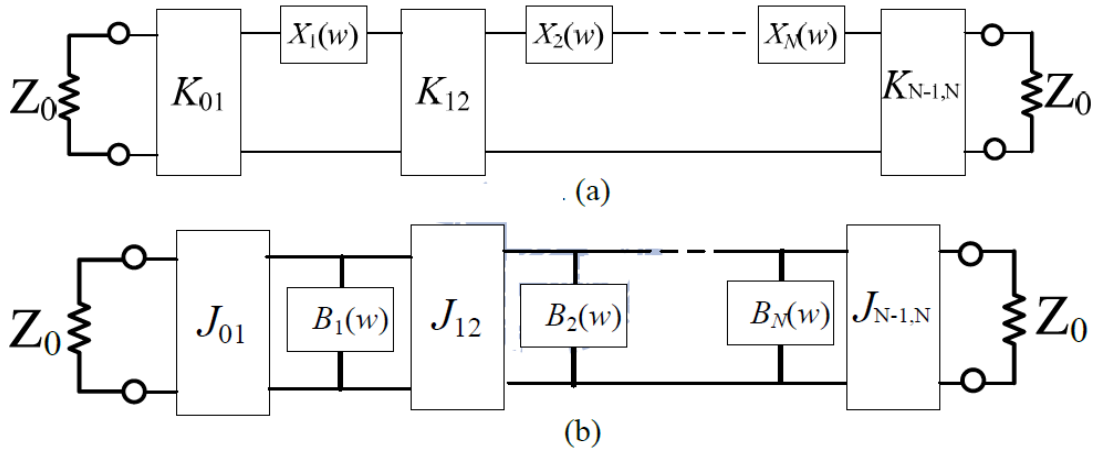
### 1.12. Impedance and Admittance Inverters:

In the simulation of filters, impedance and admittance inverters (1.22) are crucial components. An impedance inverter can be as simple as a quarter-wave transformer.



**Figure 1.22 Impedance inverters (a) and admittance inverters (b)**

Bandpass filters can be implemented in parallel resonant circuits (L, C) separated by admittance inverters or in series resonant circuits (L, C) separated by impedance inverters (K) by utilizing the characteristics of inverters.(J).



**Figure 1.23 Generalized bandpass filters. (a) Impedance inverters K (b), admittance inverters J.**



The following circuits (see Figure 1.24), which are commonly used for narrow band filters whose bandwidth does not surpass 10% of their central frequency, can be used to create these inverters:

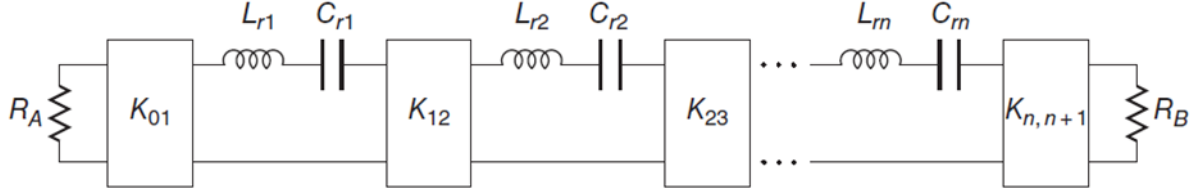


Figure 1.24 Schéma équivalent passe-bande pour les circuits à bande étroite.

## 1.12.1 Coupling Matrix Parameter:

The template, the transfer function, the equivalent circuit, and the coupling matrix can all be linked using the methods described in the earlier sections. Nevertheless, the previously presented technique is limited to direct coupling filters—that is, filters that employ Butterworth or Chebyshev functions. It is crucial to extend the preceding technique to include the couplings between non-adjacent resonators for elliptic functions. Narrow band band-pass filters can thus be linked to the localized element model [3] depicted below (Figure 1.23).

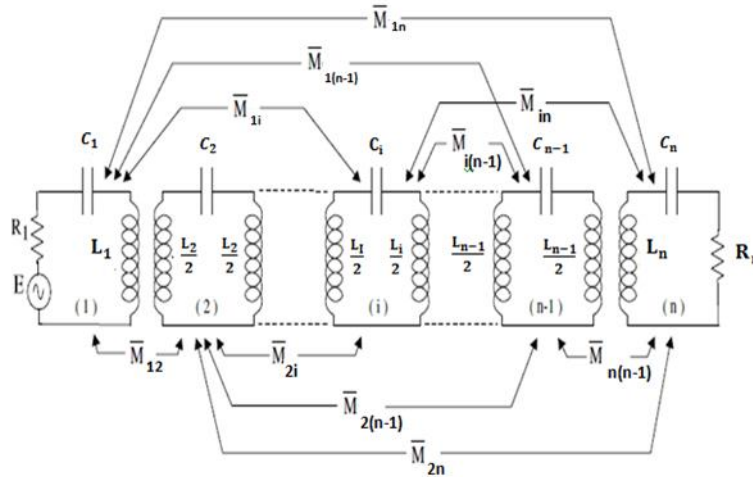


Figure 1.25 Equivalent electrical diagram for a lossless narrow-band band-pass filter, showing all possible couplings.

In the diagram, the self-inductances and capacitances define the resonators, while the mutual inductances represent the couplings between these different resonators. The

losses are not taken into account.

The coupling coefficient and quality factor can be calculated by:

$$M_{i,i+1} = \frac{FBW}{\sqrt{g_i g_{i+1}}} \quad (1.49)$$

$$Q_{en} = \frac{g_n g_{n+1}}{FBW} \quad (1.50)$$

$$Q_{e1} = \frac{g_0 g_1}{FBW} \quad (1.60)$$

FBW is the relative bandwidth.

$$Z_{01} = Z_0 = 50 \text{ ohm}$$

The characteristic impedance of the quarter-wave line  $K_{i,j}$  used as an impedance inverter between resonators  $i$  and  $j$ . [11]

$$K_{i,i+1} = \frac{Z_0}{Q_{e1} M_{i,i+1}} \quad (1.61)$$

## 1.12.2 Coupled Line Bandpass Filter:

Narrowband filters are traditionally constructed from linked line topologies. The bandwidth and the degrees of coupling between resonators are directly correlated by these structures. The figure depicts the general appearance of a microstrip bandpass filter linked in parallel. (1.26). [8]

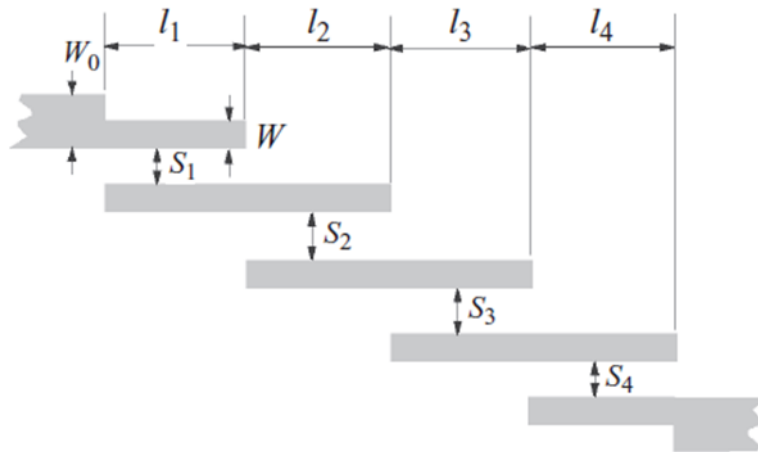


Figure 1.26: structure of a microstrip bandpass filter with coupled lines.

## 1.12.3 Band-pass filters with inverters Immittance:

The capacity to change admittance or impedance levels according to the parameters  $K$  or  $J$  that are selected. Their characteristics enable us to transform a filtering circuit into a form that is more suitable for use with microwave structures.

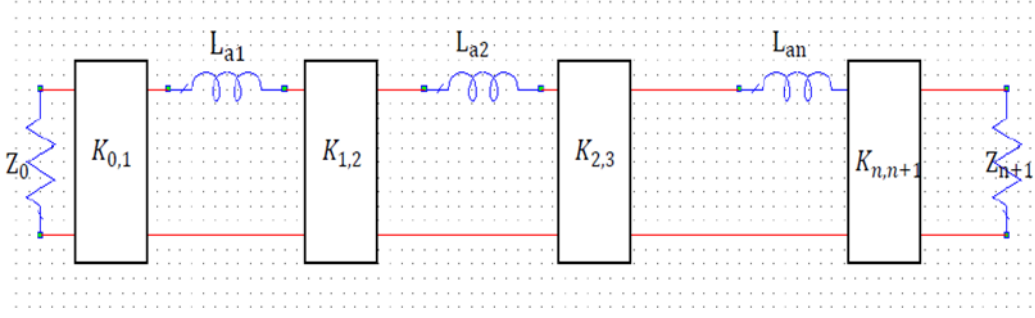


Figure 1.26 prototypes of modified low-pass filters with impedance inverters installed.

In order to convert the low-pass filter's inductances into the band-pass filter's series resonators, we get:

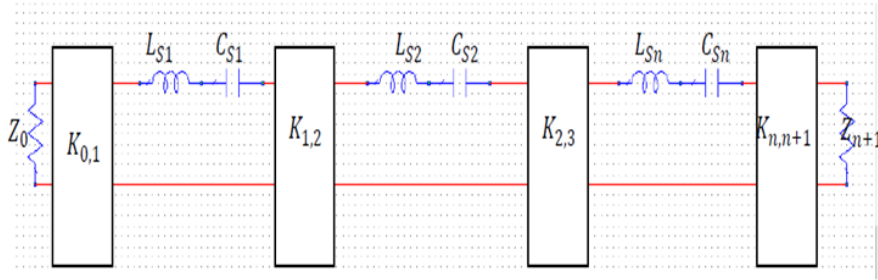


Figure 1.27 Bandpass filters using impedance inverters

Similarly, with the admittance inverters:

$$K_{0,1} = \sqrt{\frac{Z_0 \text{FBW} \omega_0 L_{s1}}{\Omega_c g_0 g_1}} \quad (1.62)$$

$$K_{i,j+1} = \frac{\text{FBW} \omega_0}{\Omega_c} \sqrt{\frac{L_{si} L_{s(i+1)}}{g_i g_{i+1}}} \quad (1.63)$$

$$K_{n,n+1} = \sqrt{\frac{\text{FBW} \omega_0 L_{sn} Z_{n+1}}{\Omega_c g_n g_{n+1}}} \quad (1.64)$$

$$C_{si} = \frac{1}{\omega_0^2 L_{si}} \mid i = 0 \text{ à } n \quad (1.65)$$

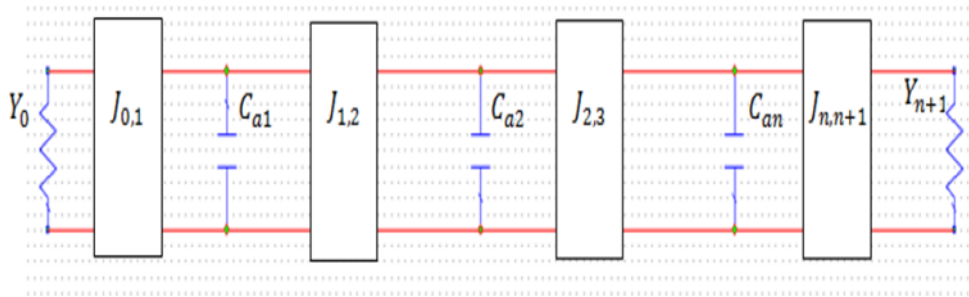


Figure 1.28 Modified low-pass filter prototypes to include admittance inverters.

$$J_{0,1} = \sqrt{\frac{Y_0 C_{a1}}{g_0 g_1}} \quad (1.66)$$

$$J_{0,1} = \sqrt{\frac{Y_0 C_{a1}}{g_0 g_1}} \quad (1.67)$$

$$J_{n,n+1} = \sqrt{\frac{C_{an} Y_{n+1}}{g_n g_{n+1}}} \quad (1.68)$$

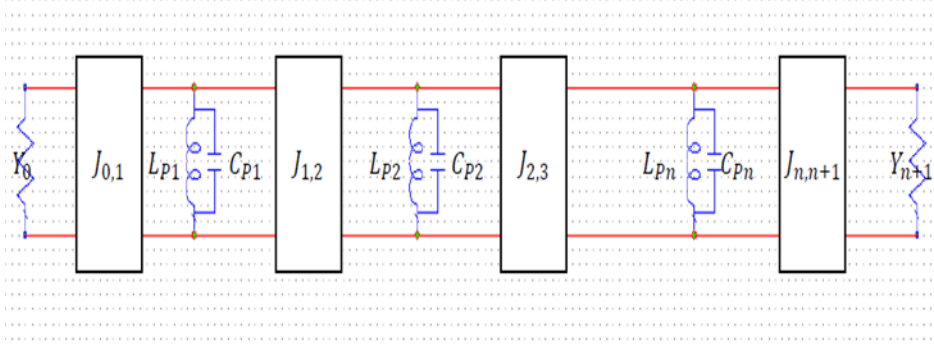


Figure 1.29 Band-pass filter using admittance inverters

$$J_{0,1} = \sqrt{\frac{Y_0 \text{FBW} \omega_0 C_{p1}}{\Omega_c g_0 g_1}} \quad (1.69)$$

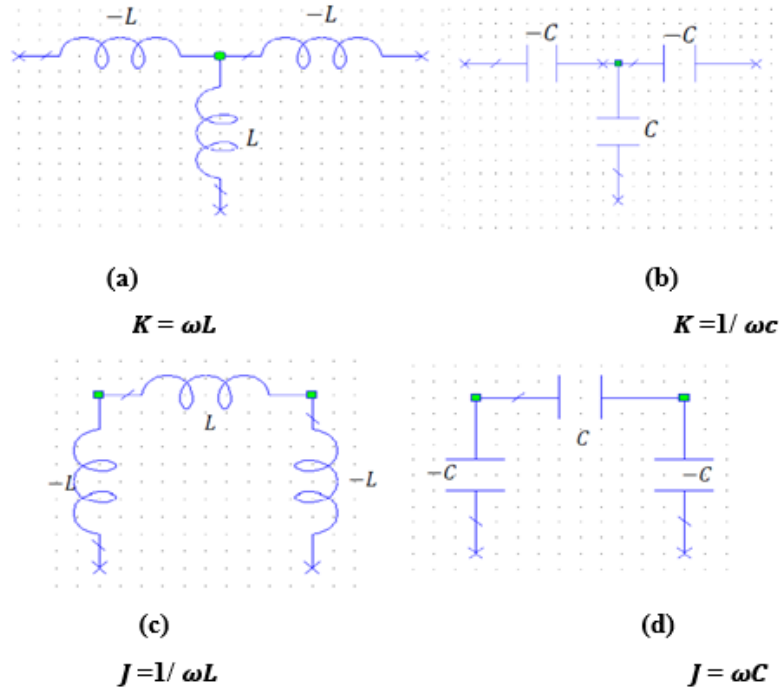
$$J_{i,j+1} = \frac{\text{FBW} \omega_0}{\Omega_c} \sqrt{\frac{C_{pi} C_{p(i+1)}}{g_i g_{i+1}}} \quad (1.70)$$

$$J_{n,n+1} = \sqrt{\frac{\text{FBW} \omega_0 C_{pn} Y_{n+1}}{\Omega_c g_n g_{n+1}}} \quad (1.71)$$

$$L_{pi} = \frac{1}{\omega_0^2 C_{pi}} \quad | i = 0 \text{ à } n \quad (1.72)$$

#### 1.12.4 Practical implementation of immittance inverters:

One of the simplest forms of inverters is a quarter-wavelength transmission line; there are other circuits that function as inverters. It can be noted that some of the inductances and capacitances have negative values. The figure shows four typical localized element inverters [7]. Figure 1.30 (a) and (b) are of interest for use as inverters K:



**Figure 1.30 immittance inverters with localized elements**

Any of these inverters can be treated as an inverter  $K$  or  $J$ . It can be shown that the inverters in figures I.23 (a) and (d) have a phase shift (the phase of  $S_{21}$ ) of  $+90^\circ$ , while those in figures I.23 (b) and (c) have a phase shift of  $-90^\circ$ . This is why the signs " $\pm$ " and " $\mp$ " appear in the matrix expressions ABCD of the immittance inverters. [7]

## 1.13 Advantages and disadvantages of microstrip filters:

Microstrip filters have a number of benefits, including low weight, low volume, conformability, and the ability to integrate microwave circuits at the filter level because their technology is developed from printed circuits. Let's also mention that this kind of filter is appropriate for low manufacturing costs due to the simplicity of their structures.

The emergence of microstrip filters in mobile communication applications can be explained by this important attribute.

The benefits of microstrip filters often come from the following:

- Simple
- Robust
- No-intrusive
- Suitable for flat and uneven surfaces
- Low-cost

Numerous benefits arise from applying faults in the mass layout, including rejection of secondary lobes and miniaturization. Microwave filters have made great use of them in their design.

### 1.14 Microstrip Technology:

Presently, filters and other passive microwave circuits are designed using microstrip technology. Indeed, it is not difficult to imagine resonators with intriguing performance and small dimensions. Figure 1.31 shows how to change the line's dimensions to configure the magnetic and electric fields for a transmission line using triple-layer technology. Figure 1.31 shows the geometry of a microstrip line (a). It is made up of a ground plane on the lower surface and a metallic strip on the upper surface of a dielectric substrate. A portion of the microstrip line's electromagnetic field lines are in the air, while the majority of them are found in the dielectric substrate Figure 1.31 (b). A microstrip line's field is a hybrid TM-TE wave rather than a pure TEM field. Nonetheless, the dielectric substrate's thickness is extremely thin in microwave applications.

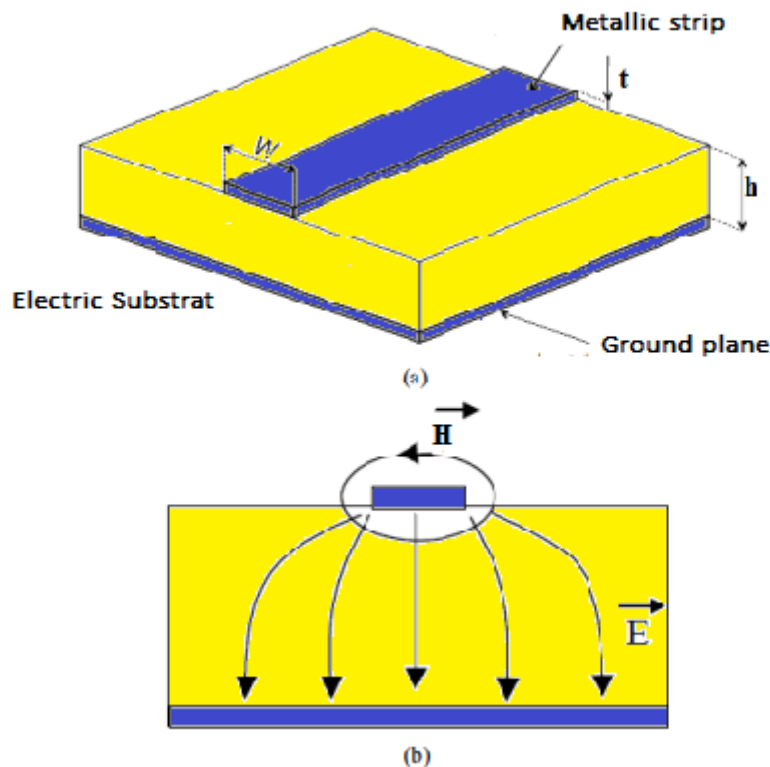


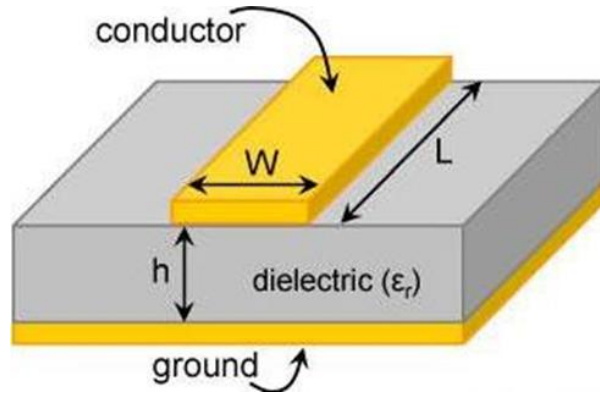
Figure 1.31: (a) Configuration of a line in microstrip technology (b) map of the fields.

The main parameters that characterize the microstrip structure are the permittivity  $\epsilon$  (often chosen to be high in order to concentrate the electromagnetic field and thus reduce radiation losses) and the geometric parameters  $W$  and  $h$  (generally  $0.1 \leq W / h \leq 10$ ).

- $W$ : line width
- Substrate thickness
- $T$ : metal thickness

### 1.14.1 Microstrip Line:

A microstrip line consists of a conductor of width  $W$ , a dielectric substrate of thickness  $d$ , and permittivity  $\epsilon_r$ . The presence of the dielectric (generally thin with  $d \ll \lambda$ ) concentrates the field lines in the region between the conductor and the ground plane, with part of it in the air region above the conductor, leading to quasi-TEM propagation modes in which dispersion occurs as a function of wavelength. [5]



**Figure 1.32: The construction of a microstrip line.**

The phase velocity and the propagation constant are given by:

$$v_p = \frac{c}{\sqrt{\epsilon_{eff}}} \quad (1.73)$$

$$\beta = k_0 \sqrt{\epsilon_{eff}} \quad (1.74)$$

With :

$$k_0 = \frac{2\pi f}{c} \quad (1.75)$$

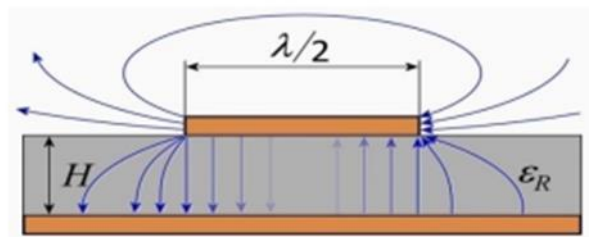
C: the speed of light ( $c \approx 3 \times 10^8 \text{ m/s}$ )

## 1.14.2 Electromagnetic field in the microstrip line:

In a microstrip line, the field lines are mainly concentrated in the dielectric between the metalized strip and the ground plane, although a small portion is also found in the air above the substrate. This implies that a microstrip line cannot support a pure TEM mode; a pure TEM wave contains only transverse components, and the propagation speed depends on the material properties (permittivity and permeability). The phase velocity of TEM fields in the dielectric  $c/\sqrt{\epsilon_r}$  differs from that in air. Waves in the microstrip line exhibit longitudinal components for the electric and magnetic fields, and the propagation speed depends on the material properties and physical dimensions of the line [12].

The distribution of the electric and magnetic fields is illustrated in the figure.

$$1 < \epsilon_{eff} < \epsilon_r$$



**Figure 1.33: Quasi-TEM Terrain Configurations**

In the quasi-static approximation, the effective relative permittivity  $\epsilon_{re}$  and the characteristic impedance of the microstrip ( $Z_C$ ) are related to the permittivity of the dielectric substrate and also take into account the effect of external electromagnetic fields.

According to:



$$Z_0 = \sqrt{\frac{L}{C}} \quad (1.76)$$

And

$$v_p = \frac{1}{\sqrt{LC}} \quad (1.77)$$

### 1.14.3 Quasi-TEM Approximation:

A portion of the field lines in microstrip are located in the air zone above the substrate, with the majority of them concentrated in the dielectric region between the conductive strip and the ground plane.

Because of this feature, the microstrip can tolerate ten waves growing.

A microstrip line's dominant mode can behave like a TEM mode and the TEM theory will apply if the longitudinal components of the fields for the dominant mode stay significantly less than the transverse components. The microstrip line can also be used with the transmission line. [6]

The quasi-TEM approximation refers to this and it is applicable to the majority of microstrip frequency working ranges.

### 1.15. Effective dielectric constant( $\epsilon_{re}$ ) and characteristic impedance( $z_c$ ):

Closed-form expression (error  $\leq 1\%$ ) for thin conductors ( $T \rightarrow 0$ ):

➤  $W/h \leq 1$ :

$$\epsilon_{re} = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2} \left\{ \left( 1 + 12 \frac{h}{w} \right)^{-0.5} + 0.04 \left( 1 - \frac{w}{h} \right)^2 \right\} \quad (1.78)$$

$$z_c = \frac{\eta}{2\pi\sqrt{\epsilon_r}} \ln \left( \frac{8h}{w} + 0.25 \frac{w}{h} \right) \quad (1.79)$$

➤  $W/h \geq 1$ :

$$\epsilon_{re} = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2} \left(1 + 12 \frac{h}{w}\right)^{-0.5} \quad (1.80)$$

$$z_c = \frac{\eta}{\sqrt{\epsilon_{re}}} \left\{ \frac{w}{h} + 1.393 + 0.677 \ln \left( \frac{w}{h} + 1.444 \right) \right\}^{-1} \quad (1.81)$$

## 1.15.1 Effective dielectric constant (error $\leq 0.2\%$ ):

$$\epsilon_{re} = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2} \left(1 + \frac{10}{u}\right)^{-ab} \quad (1.81)$$

$$a = 1 + \frac{1}{49} \ln \left( \frac{u^4 \left(\frac{u}{52}\right)^2}{u^4 + 0.432} \right) + \frac{1}{18.7} \ln \left[ 1 + \left( \frac{u}{18.1} \right)^3 \right] \quad (1.82)$$

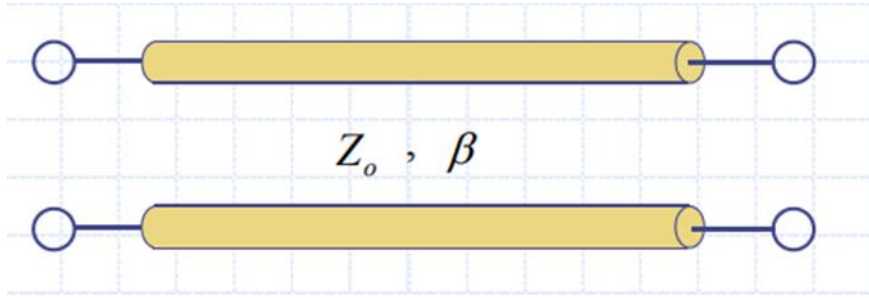
$$b = 0.564 \left( (\epsilon_r - 0.9) / (\epsilon_r + 3) \right)^{0.053}$$

(1.83)

## 1.15.2. Characteristic impedance (error $\leq 0.03\%$ ):

$$z_c = \frac{\eta}{\sqrt{2\pi\epsilon_{re}}} \ln \left[ \frac{F}{u} + \sqrt{1 + \left(\frac{2}{u}\right)^2} \right] \quad (1.84)$$

$$F = 6 + (2\pi - 6) \exp \left[ - \left( \frac{30.666}{U} \right)^{0.7528} \right] \quad (1.85)$$



- ✓ Guided wavelength:

$$\lambda_g = \frac{\lambda_0}{\sqrt{\epsilon_{re}}} \text{ ou } \lambda_g = \frac{300}{f(\text{GHz})\sqrt{\epsilon_{re}}} \text{ mm} \quad (1.86)$$

- ✓ Constante de propagation:

$$\beta = 2\pi/\lambda_g \quad (1.87)$$

- ✓ Phase velocity:

$$v_p = \frac{\omega}{\beta} = \frac{c}{\sqrt{\epsilon_{re}}} \quad (1.88)$$

- ✓ Electric length:

$$\theta = \beta \ell \quad (1.89)$$

## 1.16 Ondes de surface et modes d'ordre supérieur :

- The coupling between the quasi-TEM mode and the surface wave mode becomes significant when the frequency is above  $f_s$ .

$$f_s = \frac{c \tan^{-1} \epsilon_r}{\sqrt{2\pi h} \sqrt{\epsilon_r - 1}} \quad (1.90)$$

- Cut-off frequency  $f_c$  of the first higher-order modes in a microstrip.

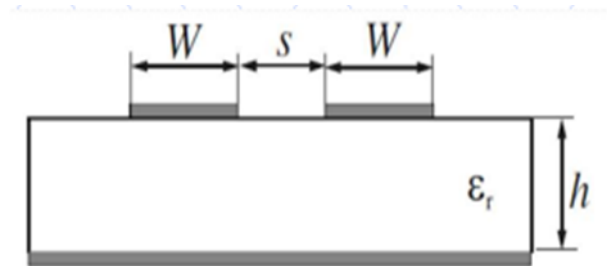
$$f_c = \frac{c}{\sqrt{\epsilon_r}(2W + 0.8h)} \quad (1.91)$$

- The operating frequency of a micro ribbon line  $< \text{Min}(f_s, f_c)$

## 1.16.1 Coupled lines:

### 1.16.1.1. Coupled Line Structure :

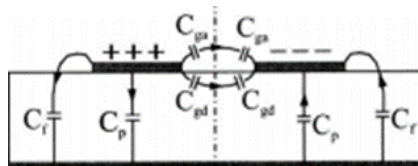
The coupled line structure supports two quasi-TEM modes: the odd mode and the even mode.



**Even and odd mode:**

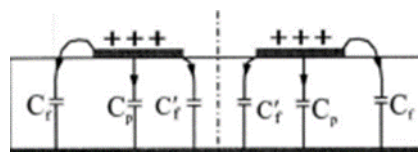
Effective Dielectric Constant ( $\epsilon_{re}$ ) and Characteristic Impedance ( $Z_c$ )

- **Odd mode:**



Electric Wall

- **Even mode:**



Magnetique wall

**Even and odd mode:**

The characteristic impedances ( $Z_c$  and  $Z_{c0}$ ) and the effective dielectric constants ( $\epsilon_{re}^0$  and  $\epsilon_{re}^e$ ) are obtained from the capacitances ( $C_0$  and  $C_e$ ).

- Even mode:

$$Z_{c0} = (c \sqrt{C_e^a C_e})^{-1} \quad (1.92)$$

$$\epsilon_{re}^e = \frac{C_e}{C_e^a} \quad (1.93)$$

- Odd mode:

$$Z_{c0} = (c\sqrt{C_0^a C_0})^{-1} \quad (1.94)$$

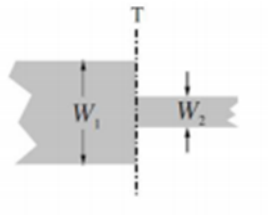
$$\epsilon_{re}^0 = \frac{C_0}{C_0^a} \quad (1.95)$$

$C_0^a$  &  $C_e^a$  are even and odd mode capabilities for the coupled microstrip line configuration with air as the dielectric.

## 1.17 Discontinuities:

- Micro ribbon discontinuities commonly encountered in the arrangement of practical filters include steps, open ends, folds, gaps, and junctions.
- The effects of discontinuities can be accurately modeled using a full-wave EM simulator or closed-form expressions and taken into account in filter designs.

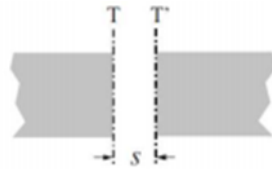
### 1.17.1. Width marches:



-Open ends:



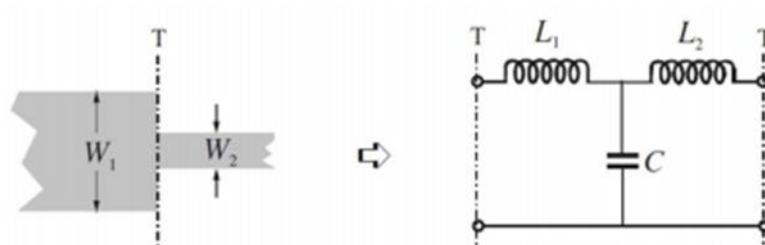
- Gaps:



- Court courses:



-Steps in width:



### 1.18. Advantages and disadvantages:

People frequently construct microwave circuits using the microstrip line. Regardless of the substrate selected, it enables a broad range of possible characteristic impedances in printed circuit technology (about  $10\Omega$  to  $200\Omega$ ). This is less true in integrated circuit technology, where their employment is restricted to impedances below roughly  $70\Omega$  due to the substrate's few millimeter thickness. The losses are increasing as the ribbon's width decreases.

Connecting parts in series is still simple, but placing parts in parallel is more difficult and needs vias to guarantee that the bottom surface of the component is connected to the ground plane. Metalized holes, also known as metal vias, have a noticeable impact on the electrical performance of the circuit, and their parasitic effects need to be considered. Even with these minor shortcomings, this technology is still in high demand. These benefits frequently exceed the drawbacks.

### 1.19 Conclusion

In this chapter, we presented the design method for the different types of microwave filters and the microstrip technology. In fact, the optimal normalized low-pass filter can be used to derive the various categories. However, because a discontinuous attenuation cannot be achieved, the perfect low-pass filter is not physically feasible. Finding a transfer function that most closely resembles the properties required by the template of the standardized low-pass filter reference is thus a first step. The most popular solutions Butterworth, Chebyshev, and elliptic type approximation functions—have been explained. To obtain an adequate rejection rate using the Butterworth approximation, a very high filter order is necessary, which invariably results in a sizable amount of losses in the real filter. Regarding the Chebyshev approximation, it is more widely utilized because to its ease of implementation and ability to provide, depending on the permitted ripple, a greater rejection in the stopband for an equivalent order. Apart from these traditional filters, elliptical approximations with rippling in the passband and stopband exist. These filters' attenuated band contains discrete frequencies where transmission zeros are present, which enables large rejection levels with a constrained order. These filters work well, but the related electrical schematic and synthesis can be difficult to accomplish, which adds complexity to the design.

We associate an electrical schematic made up entirely of localized elements of kinds L and C (capacitances and inductances) with the selected mathematical function. Mathematical formulas are used to determine the circuit elements. Next, we alter the circuit elements' impedance and frequency to provide the required filtering function (band-pass, band-stop, etc.).

The benefits and drawbacks of microstrip technology were discussed in this chapter, along with some of its general properties including the effective dielectric constant ( $\epsilon_{eff}$ ) and characteristic impedance  $Z_0$ . Additionally, models with various discontinuities are offered.

# Chapter Two



### 2.INTRODUCTION:

Deep learning has gained increasing appreciation in both academic and industrial fields over the past decades. Various domains, such as computer vision, pattern recognition, and natural language processing, have witnessed the tremendous power of deep networks. However, current studies on deep learning primarily focus on datasets with balanced class labels. [13]

### 2.2. Convolutional Neural Networks (CNNs):

Convolutional networks, also known as Convolutional Neural Networks (CNNs) (LeCun, 1989), are a class of deep learning neural networks, that are widely used for tasks like image recognition, natural language processing, and other applications where detecting spatial or temporal patterns in data is crucial. CNNs are built to automatically and adaptively learn spatial feature hierarchies using backpropagation, leveraging various components such as convolutional layers, pooling layers, and fully connected layers. [14]

#### 2.2.1How a CNN Works

Convolutional Neural Networks (CNNs) are currently the most competitive models for classifying images. Referred to by the acronym CNN, they have two clearly defined parts. The input is an image with numerical values in the form of a pixel matrix, arranged spatially: width, height, and depth (dimensions) as shown in Figure 1. The image has 2 dimensions (2D) for grayscale images. A color image is represented by a third dimension, depth (3D), which corresponds to the fundamental colors [Red, Green, Blue] or RGB.

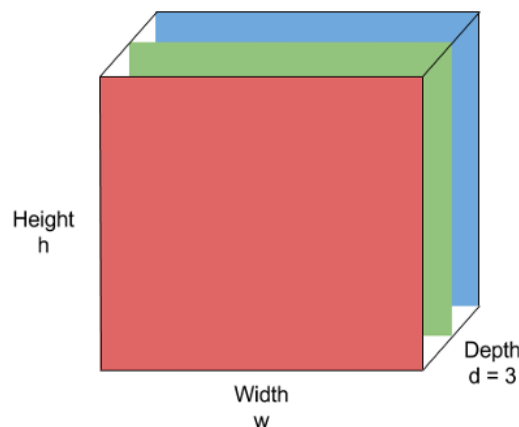


Figure 2.1: Image arranged spatially: width, height, and depth.

## Chapter II : Overview Of CNN

The convolutional part is the first section of Convolutional Neural Networks (CNNs), which functions as a feature extractor for images. Represented as a matrix, it passes through a sequence of filters: convolution layers and pooling layers, creating new matrices (images) called convolution maps. At the end, the convolution maps are concatenated into a feature vector, called the CNN code.

This CNN code is then fed into the second part, known as Fully Connected layers. The main role of this section is to combine the features from the CNN code to classify an image.

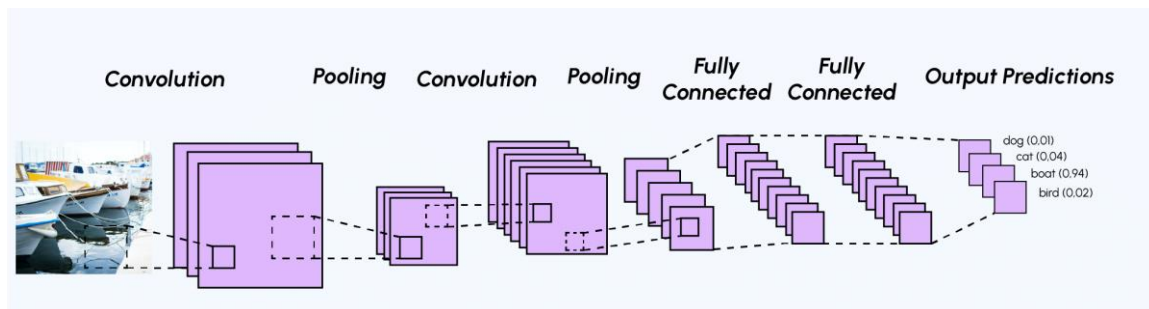


Figure 2.2: How CNN works

A CNN architecture (**Figure 2**) is composed of a stack of independent processing layers:

- The **Convolutional layer (CONV)**, which processes data from a receptive field.
- The **Pooling layer (POOL)**, which compresses information by reducing the size of the intermediate image (often through subsampling).
- The **Correction layer (ReLU)**, commonly referred to as 'ReLU' in reference to the activation function (Rectified Linear Unit).
- The **Fully Connected layer (FC)**, which is a perceptron-type layer.
- The **Loss layer (LOSS)**.

### 2.2.2 Convolutional Layer (CONV)

The convolutional layer is the basic building block of a CNN. Three parameters determine the volume of the convolutional layer: depth, stride, and padding.[14]

- a- **Layer Depth:** The number of convolutional kernels (or the number of neurons associated with the same receptive field)

## Chapter II : Overview Of CNN

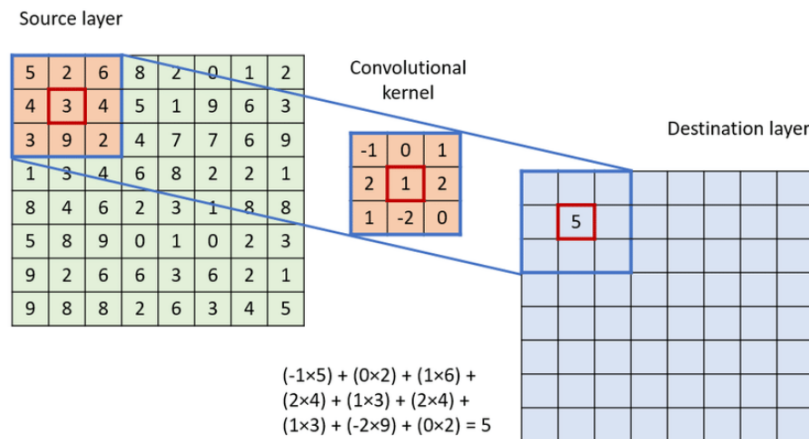


Figure 2.3: Principle of kernels

- b- **Stride:** Controls the overlap of receptive fields. The smaller the stride, the more receptive fields overlap, and the larger the output volume will be.
- When using a stride of 1 (stride=1), it means we are moving our filter (3x3) by 1 pixel at a time (Figure 2.4)

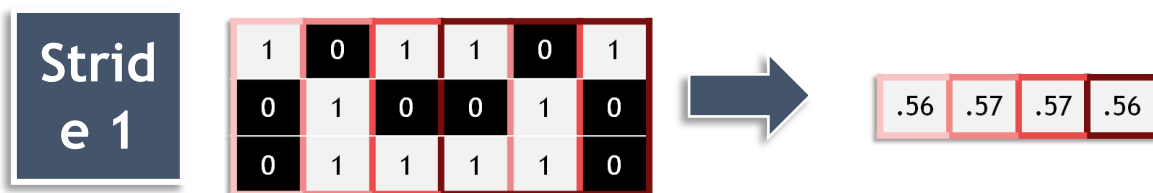


Figure 2.4: An example of a stride of 1 pixel

In this image, you can see an example of a 3x3 filter (kernel) applied to an input matrix with a stride of 1. The filter slides over the input, moving 1 pixel at a time, as illustrated by the highlighted squares. After the convolution operation is performed at each position, the resulting values (e.g., 0.56, 0.57) are output as part of a feature map. [15]

Here's what happens

- The filter starts at the top-left corner of the matrix, covering the first 3x3 block. It calculates the result (e.g., 0.56) for that position.
- The filter moves 1 pixel to the right, covering the next 3x3 block, and calculates the next output value (e.g., 0.57).
- This process continues across the entire input matrix, moving 1 pixel at a time, producing the output feature map.

## Chapter II : Overview Of CNN

- This visualization shows how stride 1 ensures that the filter moves incrementally by 1 pixel, processing each overlapping region of the input matrix.
- When using a stride of 2 (stride=2), it means we are moving our filter (3x3) by 2 pixel at a time (Figure 2.5)

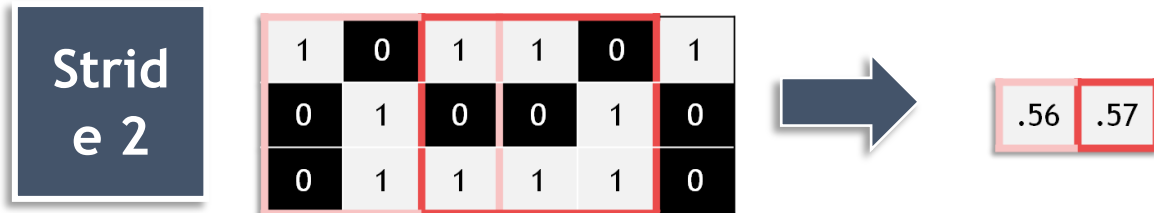


Figure 2.5: An example of a stride of 2 pixels

- c- **Padding:** Sometimes, it is useful to add zeros to the boundary of the input volume. The size of this 'zero-padding' is the third hyperparameter. Padding helps control the spatial dimension of the output volume. If we want enough data to make sure all pixels are used in convolution, or if we want the resulting image to be the same size as our input image, we can do something called padding. A quick and easy way to do this is called “zero” padding, where we add a border of zeros around our image. This can be sufficient in many cases, but in some where the image is small, it can have an impact on the convolution.[14][16]

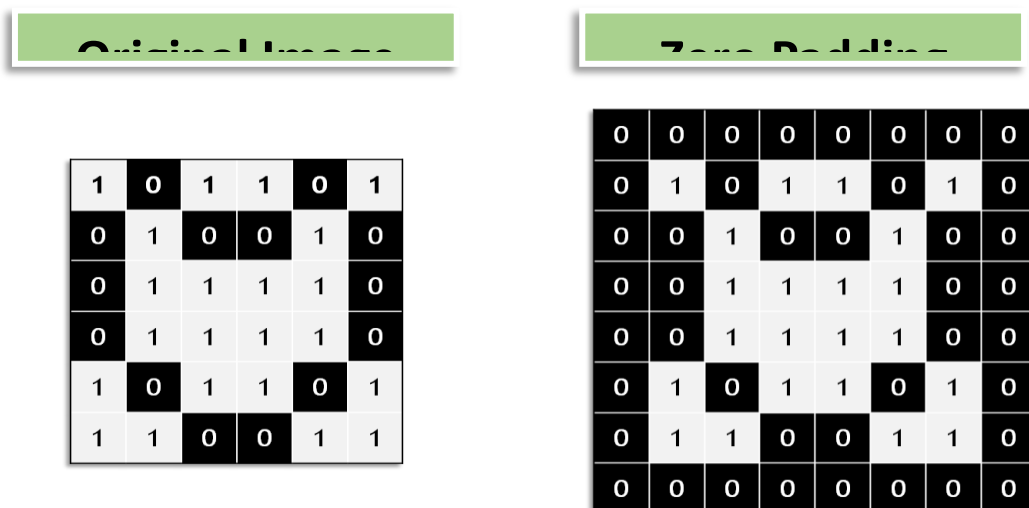


Figure 2.6: zero padding

There are many other types of padding too. Many frameworks have “mirror padding” that instead duplicates the top and bottom rows, and the left and right columns.

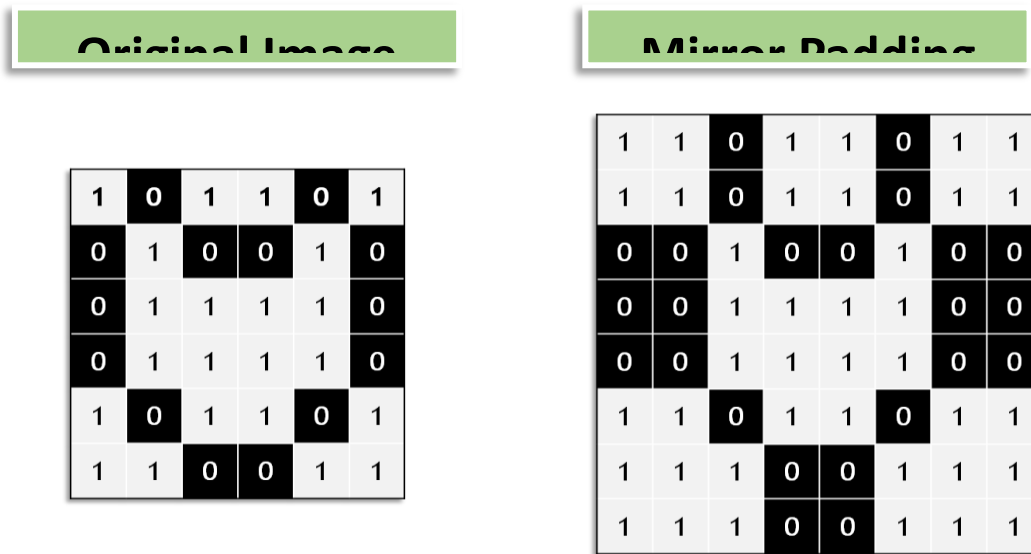


Figure 2.7: mirror padding

### 2.2.3. Pooling layer:

This type of layer is often placed between two convolutional layers: it receives multiple feature maps as input and applies the pooling operation to each of them. A pooling layer acts as a reduction layer. It divides the image into blocks and keeps only the maximum value from each block. This reduces the size of the image while preserving the most important features. The output consists of the same number of feature maps as the input, but they are significantly smaller. There are several key types of pooling (max, average and global pooling).[16][17]

The most common one is Max Pooling where we take the largest value in the window and discard the rest. This trick is especially useful when working with large images because it's a way to shrink images down to a smaller size, and smaller images mean less computation, which is quicker and cheaper.

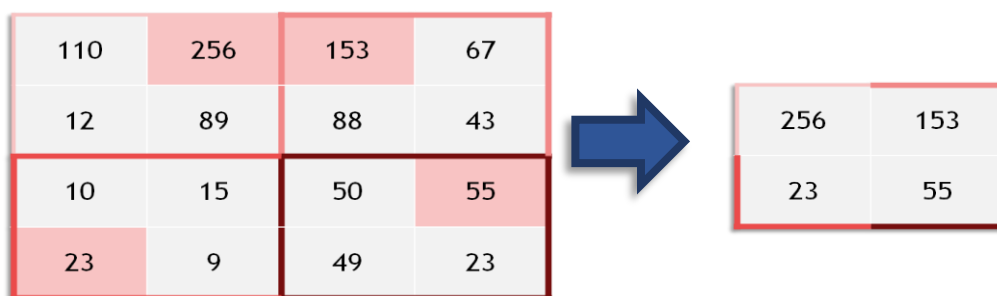


Figure 2.8: max pooling

### 2.2.4 The Correction layer (ReLU):

ReLU is an **activation function** used in neural networks, and it is one of the most commonly used activation functions, particularly in deep learning models like CNNs. It is mathematically defined as:

$$f(x) = \max(0, x)$$

This means that if the input  $x$  is positive, the function returns  $x$ ; if the input is negative or zero, it returns 0. The ReLU function is computationally simple and introduces non-linearity into the network, allowing it to learn and model more complex patterns.



Figure 2.9: Relu function

### 2.2.5 Flattening (Aplatissement)

After applying convolution and max-pooling operations, the process of flattening involves taking the elements from the pooled feature maps and converting them into a one-dimensional vector. This flattened vector, often referred to as the CNN code, becomes the input for the fully connected layer.

Flattening is a crucial step in transitioning from the spatial structure of the feature maps to the fully connected layers, where high-level reasoning or classification occurs. This transformation allows the model to use the learned features for tasks such as image classification, detection, or regression. [17]

### 2.2.6. The Fully connected layer (FC):

After several convolutional and max-pooling layers, higher-level reasoning in the neural network is performed through fully connected layers. The neurons in a fully connected layer have connections to all the outputs from the previous layer. Their activation

functions are thus computed by performing a matrix multiplication followed by a bias shift.

This layer is responsible for combining features extracted by the previous convolutional layers and determining the final classification or prediction by processing the features in a more holistic manner.[16]

### 2.2.7. Loss Layer (LOSS):

The loss layer specifies how the training process penalizes the difference between the predicted and actual signals. It is typically the last layer in the network. Various loss functions, suited to different tasks, can be used in this layer.

The Softmax function is often employed to calculate the probability distribution over the output classes. It ensures that the output values are converted into probabilities that sum to 1, which is useful for classification tasks. For instance, in a multi-class classification problem, Softmax helps the network output the most likely class for a given input by assigning a probability to each possible class.

- Common Loss Functions:
  - Mean Squared Error (MSE): Used in regression tasks where the goal is to minimize the squared difference between predicted and actual values.
  - Cross-Entropy Loss: Commonly used in classification tasks, especially when combined with the Softmax function. It measures the performance of the model by comparing predicted probabilities to the actual class labels. The loss increases as the predicted probability diverges from the actual label.
  - Hinge Loss: Used for tasks like support vector machines (SVM), particularly for binary classification.

By minimizing the loss function during training, the network learns to adjust its weights, reducing the error and improving accuracy on future predictions.

- After the flattening step in a Convolutional Neural Network (CNN), where the image is converted into a one-dimensional vector, the subsequent layers operate similarly to those in a fully connected neural network (ANN). The fully connected layers that follow treat the data just like in a traditional ANN, where each neuron in a layer is connected to every neuron in the previous layer, and each connection has a weight that is learned during training.

So, while CNNs use convolutional layers to extract features from images, the fully connected layers at the end of the network process the data in a manner similar to a traditional ANN. This allows the network to combine and use the extracted features to make final predictions.

### 2.3 ANN (Artificial Neural Networks):

#### 2.3.1 Introduction:

Artificial Neural Networks (ANNs) are computational algorithms designed to simulate the workings of biological neural networks, such as the human brain. Just as biological neurons process and transmit information, ANNs are composed of interconnected units that work together to handle complex data processing tasks and generate meaningful results. They are used to address a wide range of problems, from image recognition and natural language processing to time series forecasting.[18]

#### 2.3.2 Architecture of Artificial Neural Networks (ANNs):

The architecture of an Artificial Neural Network (ANN) refers to the structure and arrangement of its components. This includes the layers, the types of neurons, and the connections between them. Here's a breakdown of the typical components of an ANN architecture:

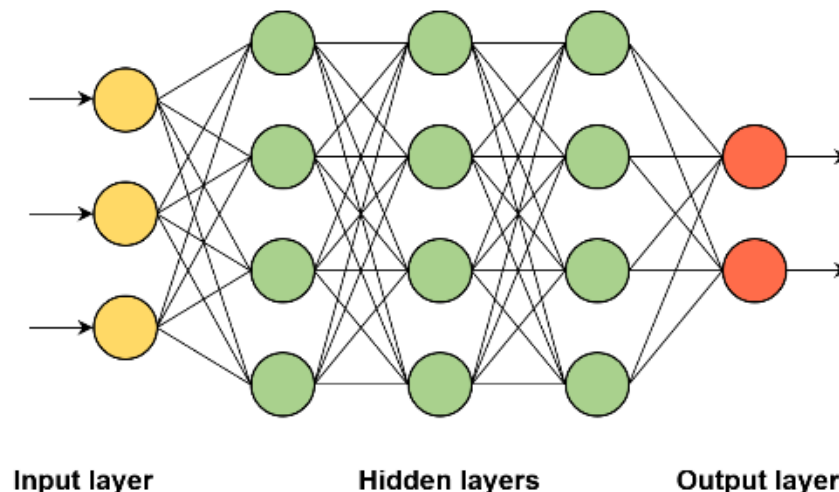


Figure 2.10 : A simple neural network

#### a. Layers:

- Input Layer: The first layer of the network where data is fed into the network. Each neuron in this layer represents a feature of the input data.



## Chapter II : Overview Of CNN

---

- Hidden Layers: One or more intermediate layers where computations are performed. These layers contain neurons that apply activation functions to their inputs to transform and extract features from the data. The complexity of the network often increases with the number of hidden layers.
- Output Layer: The final layer that produces the network's prediction or classification. The number of neurons in this layer typically corresponds to the number of classes or the dimensionality of the output.

### **b. Neurons:**

Each neuron in a layer is connected to every neuron in the previous layer, and these connections have associated weights that are adjusted during training.

Neurons compute a weighted sum of their inputs, add a bias term, and apply an activation function to produce their output.

### **c. Activation Functions:**

Activation functions introduce non-linearity into the network, allowing it to model complex relationships in the data. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh.[19]

## Chapter II : Overview Of CNN

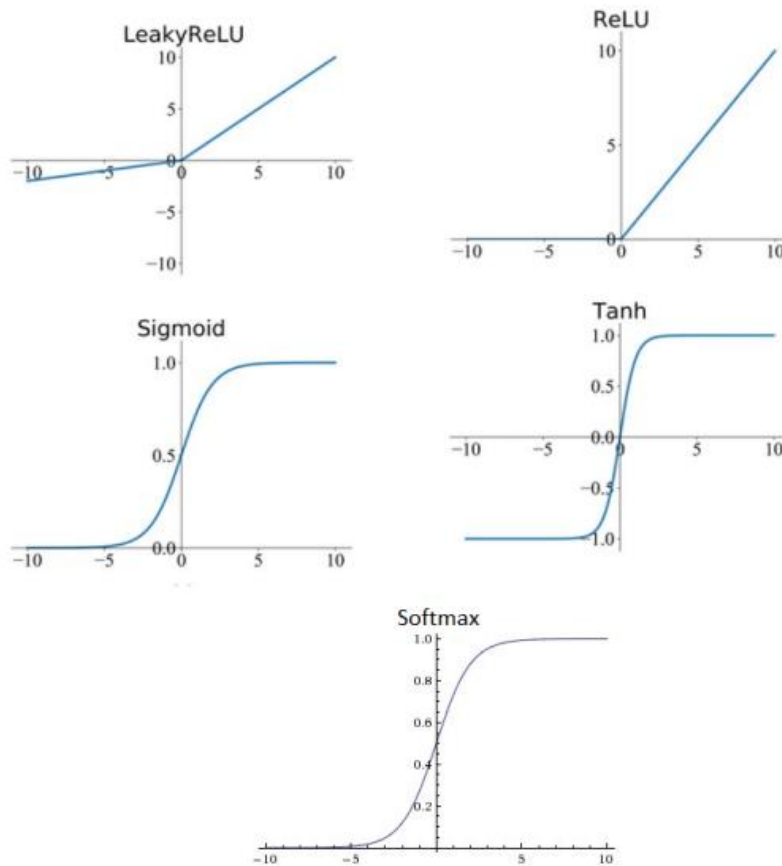


Figure 2.11: Activation functions

- **Sigmoid Function:** The sigmoid function generates an S-shaped curve that converts any input into a value between 0 and 1. It is frequently used in binary classification tasks where the goal is to predict one of two possible outcomes.
- **ReLU Function:** The ReLU (Rectified Linear Unit) function outputs the input value if it is positive; otherwise, it returns zero. Known for its computational efficiency, ReLU is widely utilized in various neural network architectures.
- **Tanh Function:** The Tanh (Hyperbolic Tangent) function is akin to the sigmoid function but maps inputs to a range between -1 and 1. It is commonly employed in neural networks for both regression and classification tasks.
- **Softmax Function:** The Softmax function converts inputs into a set of values that indicate the probabilities of each possible outcome in a multi-class

## Chapter II : Overview Of CNN

---

classification scenario. The outputs of the Softmax function sum up to one, making it ideal for predicting the likelihood of different classes.

### d. Weights and Biases:

Weights determine the strength of the connection between neurons. Biases are additional parameters that allow the activation function to shift, providing more flexibility to the model.

During training, weights and biases are adjusted through optimization algorithms to minimize the prediction error.

### e. Forward Propagation:

This is the process of passing input data through the network to obtain a prediction. Each neuron performs a calculation based on its inputs, weights, and activation function.

The network starts by initializing weights and using the inputs,  $x$ , as the initial activations. Each neuron then computes the weighted sum of the activations from the previous layer and adds a bias term. This linear combination,  $z(W, b)$  is then processed through an activation function,  $a(z)$  to produce the activations for the neurons in the next layer. This process is repeated across all layers, ending with the output layer where the final neuron outputs represent the network's prediction.

Mathematically, this can be expressed as:

$$z_i^k(W, b) = \sum_{j=1}^{n^{(k-1)}} (w_{i,j}^k * a_j^{(k-1)}) + b_i^k$$
$$a_i^k(z) = g(z_i^k)$$

Where  $a_i^k$  is the output of the neuron  $i$  of layer  $k$  and  $n^{(k-1)}$  is the number of neurons in the previous layer.  $w_{i,j}^k$  is the weight of the connection between neuron  $j$  in layer  $(k-1)$  and neuron  $i$  in layer  $k$ .  $g(x)$  is the activation function.  $b_i^k$  is the bias of neuron  $i$  in layer  $k$ .

### f. Backpropagation:

## Chapter II : Overview Of CNN

---

After forward propagation, backpropagation is used to update the weights and biases by calculating the gradient of the loss function with respect to each parameter and adjusting them to minimize the error.

The backpropagation algorithm can be decomposed in the following steps:

- Feed-forward computation
- Backpropagation to the output layer
- Backpropagation to the hidden layers
- Parameters updates

The algorithm stopped when the value of the error function has reached a satisfactory level of smallness.

The parameter update is performed in the following manner:

$$w_{i,j}^k = w_{i,j}^k - \alpha * \frac{\delta E}{\delta w_{i,j}^k}$$

$$b_i^k = b_i^k - \alpha * \frac{\delta E}{\delta b_i^k}$$

Where  $\alpha$  is a hyperparameter of the network, known as the learning rate, that determines how rapidly the parameters are updated. If the learning is too big, the optimal values will be overshoot. In contrast if it is too small, the convergence will require too many iterations. To execute the preceding equations, it is essential to apply the chain rule. Hence, in a simplified manner, the steps will be as follows:

$$\frac{\delta E}{\delta z} = \frac{\delta E}{\delta a} * \frac{\delta a}{\delta z}$$

$$\frac{\delta E}{\delta w} = \frac{\delta E}{\delta z} * \frac{\delta z}{\delta w}$$

Where  $\frac{\delta a}{\delta z}$  is the derivative of the activation function

Finally:

$$\frac{\delta E}{\delta w_{i,j}^k} = a_j^{k-1} * \frac{\delta E}{\delta z}$$

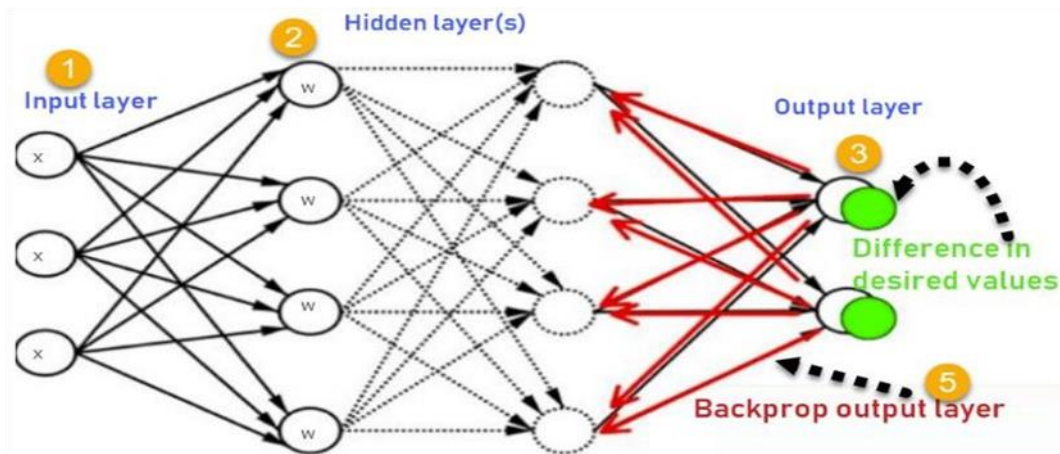


Figure 2.12: Operation of the Back propagation Algorithm [20]

### g. Optimization Algorithms:

Algorithms like gradient descent, Adam, and RMSprop are used to optimize the weights and biases by minimizing the loss function during training

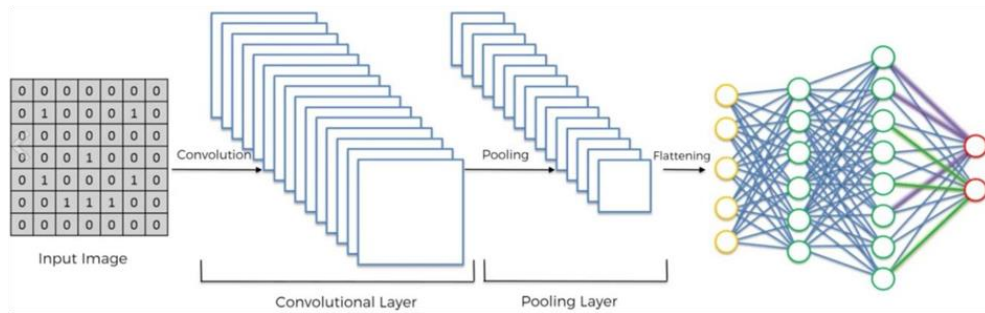
### 2.3.3. The Relationship between the Convolutional Layer and the Fully Connected Layer:

Let's take the example of a human face. The convolutional layer in a neural network can identify individual features such as the nose, ears, and eyes. However, while it detects these features, it doesn't necessarily understand where they should be positioned relative to each other. This is where the fully connected layers come into play. These layers combine the detected features and process them in a way that creates a more complex and holistic model of the input data. By doing this, the fully connected layers allow the network to better understand spatial relationships between features and make more accurate predictions—such as classifying the input as a human face.

In essence, the convolutional layers serve as feature extractors, identifying important patterns in the input (such as edges or shapes), while the fully connected layers use these features to make final decisions or classifications. The fully connected layers are able to give the network greater predictive power by integrating the abstract features learned by the convolutional layers into a coherent structure that is meaningful for the task at hand.

The fundamental difference between a Convolutional Neural Network (CNN) and an Artificial Neural Network (ANN) lies primarily in this preprocessing stage. CNNs apply convolutional filters to the input data, which allows them to automatically extract spatial

hierarchies of features. ANNs, on the other hand, typically require manual feature extraction and do not inherently understand spatial structures in the data. CNNs are, therefore, particularly effective for tasks like image recognition, where the spatial relationships between features are crucial, while ANNs might be more suited to simpler tasks that do not involve spatial data. [14]



**Figure 2.13: The Relationship between the Convolutional Layer and the Fully Connected Layer (ANN) [21]**

### 2.4 Conclusion:

Convolutional Neural Networks are a powerful class of models for analyzing data with spatial or temporal patterns. Their ability to automatically learn complex features and patterns has made them the go-to method for a wide range of applications, from image recognition to more specialized domains like waveguide filter modeling and antenna design. As deep learning continues to advance, CNNs remain at the forefront of innovation in artificial intelligence.

While CNNs can be considered a specialized type of ANN, designed to handle spatial data, the main difference is how they process and extract features. ANNs are more general-purpose and work well for simpler tasks, whereas CNNs excel in handling visual or sequential tasks by automatically extracting and learning hierarchical patterns in data. Both play crucial roles in modern deep learning, with their usage determined by the type and complexity of the problem being addressed.

# Chapter Three

### Chapter III Experimental results

#### 3.1 Introduction:

Microwave engineering principles are combined with deep learning algorithms to provide an advanced technique called deep learning for microwave modeling and design. By utilizing neural networks and massive datasets, it allows engineers to create precise models and optimize the design of microwave components and systems.

#### 3.2 Methodology:

The power of deep learning is combined with the features of HFSS, a popular electromagnetic simulation program, and the adaptability of the Python programming language in Deep Learning for Microwave Modeling and Design.

Microwave engineers frequently utilize HFSS as a tool to simulate, analyze, and solve complicated electromagnetic problems involving high-frequency electromagnetic fields. It offers a wide range of tools for modeling and creating different microwave systems and components.

Conversely, Python is a well-liked programming language that provides a large selection of deep learning modules and frameworks. Deep learning model training is made easier with libraries like PyTorch and TensorFlow, which offer effective neural network and algorithm implementations.

Engineers can combine the benefits of both tools when using HFSS and Python for deep learning in microwave modeling and design. This is a quick synopsis of the procedure:

##### ➤ Data preparation:

By simulating the behavior of microwave components or systems under various situations, HFSS can be utilized to create training data. It is possible to extract and store the simulation results as datasets, including S-parameters and electromagnetic field distributions.

##### ➤ Deep learning model development:

Engineers can create deep neural network models that can learn from the simulated data by using Python and deep learning packages. These models can be made to represent intricate linkages, like the performance metrics of the microwave system or device, between the input parameters and the intended output.



### ➤ Training the model:

The deep learning model is trained using the prepared datasets. By analyzing the data's input-output patterns, the model learns and modifies its internal parameters to reduce prediction mistakes. This procedure entails feeding the model with data, computing the loss, and applying optimization algorithms such as stochastic gradient descent to optimize the model.

### ➤ Model evaluation and refinement:

Using validation datasets, the model's performance is assessed following training. The accuracy, efficiency, and generalizability of the model are examined by engineers. The model can be improved if needed by modifying its training procedure, hyper parameters, or architecture.

### ➤ Model deployment and utilization:

The trained model can be used to forecast the behavior of novel microwave systems or components when it satisfies the required performance standards. Without requiring lengthy simulations, the model can estimate crucial parameters like S-parameters, resonance frequencies, or radiation patterns by giving input parameters.

In general, engineers can improve the effectiveness, precision, and automation of the design process, resulting in quicker and more creative microwave systems, by integrating HFSS with Python for deep learning in microwave modeling and design.

## 3.3 Data preparation

Data preparation using HFSS (High-Frequency Structure Simulator) involves generating and extracting relevant simulation data to create datasets for deep learning applications in microwave modeling and filter design. In this project, we focused on analyzing the performance of a microstrip coupled line, following a series of structured steps.

First, we created the 3D geometry of the microstrip coupled line using HFSS's drawing tools. This involved designing the rectangular patch and feeding line, ensuring precise dimensions and positions based on the design specifications.

Next, we assigned the appropriate material properties to the various components of the antenna. We selected a dielectric material from the HFSS material library that closely

## Chapter III : Experimental Results

matched the substrate material used in our physical design. This step is essential for producing accurate simulation results.

A microstrip filter is designed to meet the specifications:

Center frequency	4.5 GHz
Passband bandwidth	1 GHz
Passband return loss	$< -15$ dB
Rejection bandwidth (35 dB)	$> 2$ GHz
Dielectric Constant ( $\epsilon_r$ )	10.5
Dielectric Height (h)	0.6 mm

**TableIII.1: Specifications of microstrip**

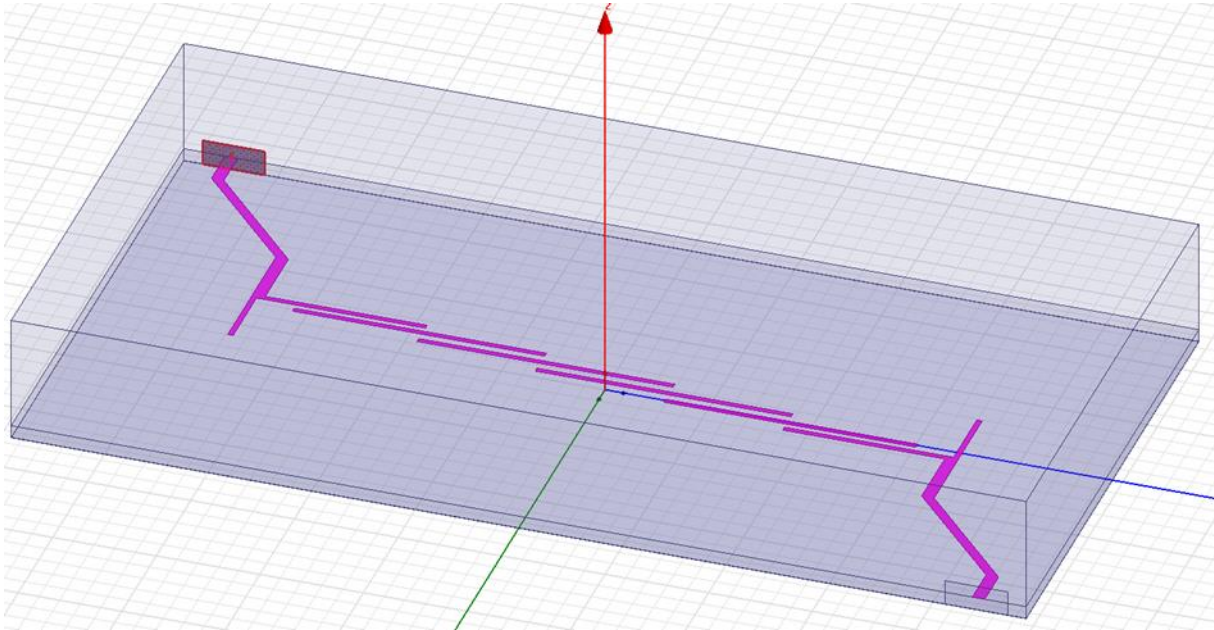
To specify how the microstrip coupled line antenna interacts with its environment, we put up the boundary conditions. To prevent reflections, we erected radiation limits to stop electromagnetic waves from traveling away from the structure.

With physical parameter presented in this table:

L1	5.889
L2	6
L3	5.835
L4	2.13
L5	3.76
S1	0.303
S2	0.458
S3	0.474

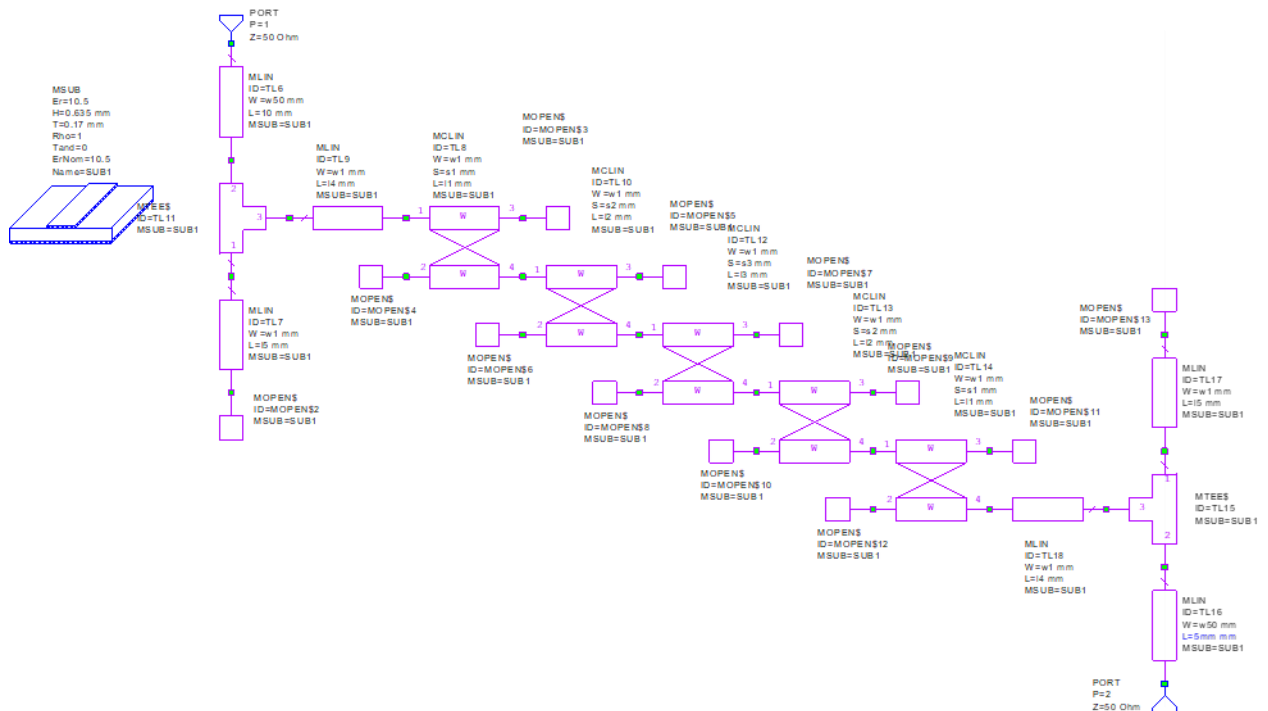
**TableIII.2: Physical parameter of microstrip**

## Chapter III : Experimental Results



**Figure III.1: Microstrip Design simulation**

This is example using the AWR software, we plotted schema of the microstrip filter by inputting all the data from the table above.



**Figure III.2: Microstrip coupled line structure.**

## Chapter III : Experimental Results

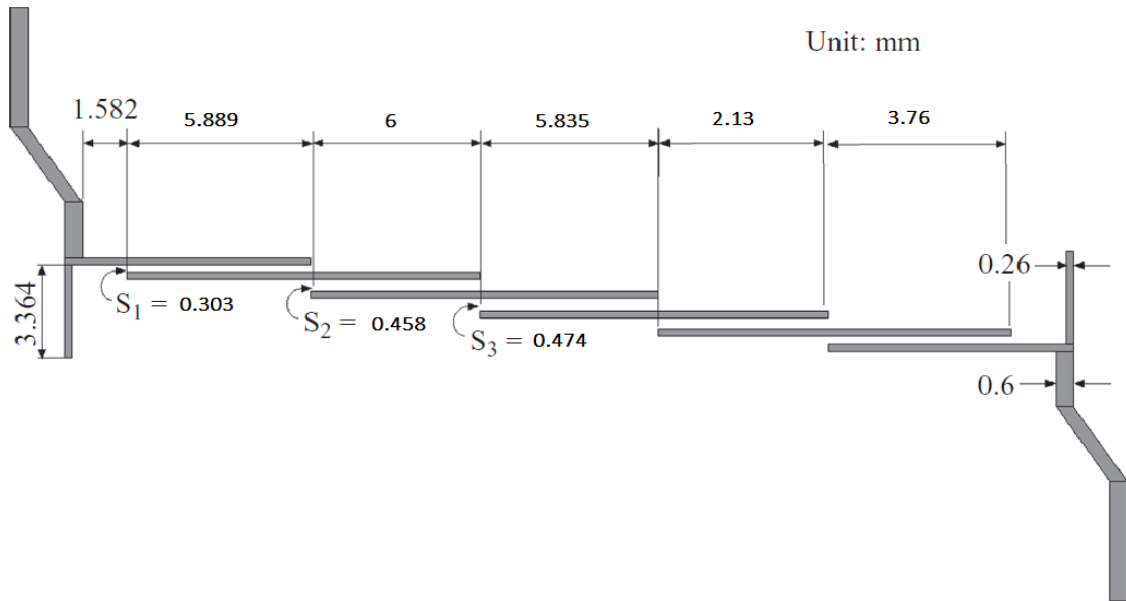
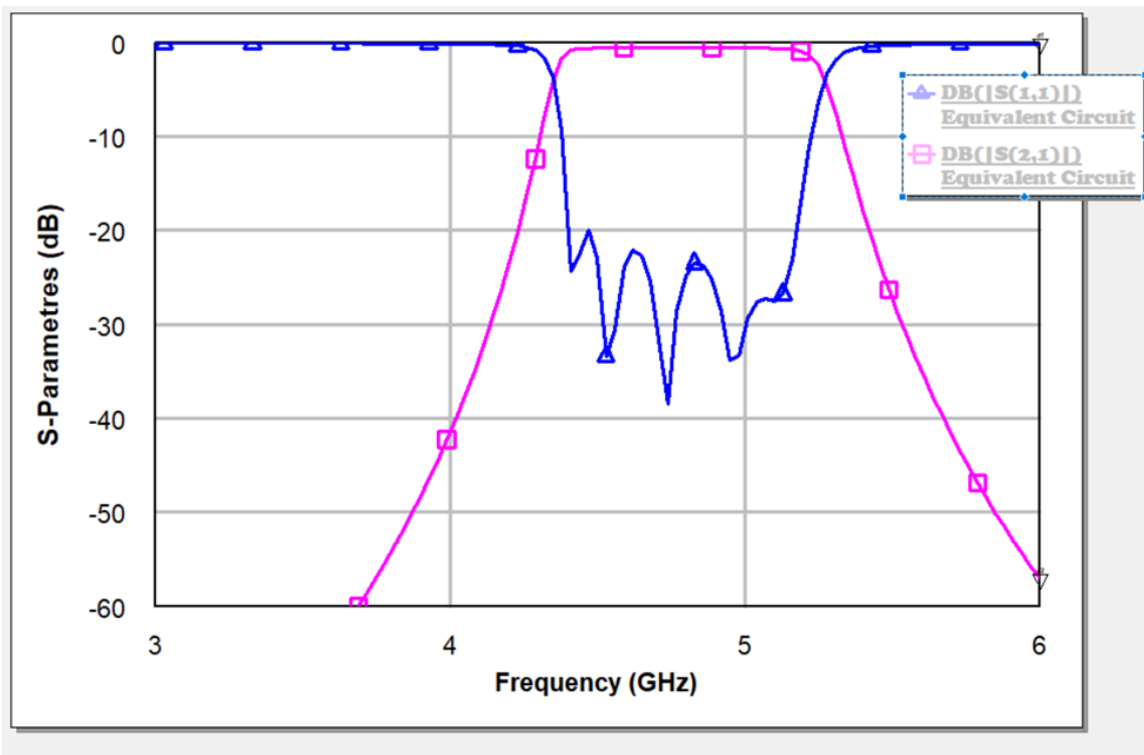


Figure III.3: Layout of the designed edge-coupled microstrip filter on Substrate with a relative dielectric constant of 10.5.

These results are after simulation in the AWR program:



FigureIII.4: Plot of S11&S21 as function with frequency.

## Chapter III : Experimental Results

---

First, we identified the parameters to be varied in our analysis. These could include geometric dimensions, material properties, or any other variables affecting the structure's behavior.

Next, we defined variables to represent the parameters. For this analysis, we created the variables "L1," "L2," "S1" and "S2" & Frequency and set their initial values.

We began the simulation run. HFSS automatically ran several simulations, varying the swept parameter value in each. For example, we utilized count equals to 5 counts in this instance, and the thickness was  $0.635 \pm 0.0254$  mm, with a relative dielectric constant of  $10.5 \pm 0.25$ . We limit it to 0.2 mm for both the narrowest line width and the tightest gap between lines. To fit the filter within the necessary size, the first and last resonators are bent by a length of L5. Two 50- $\Omega$  tapped wires realize the input and output (I/O).

The effect of the microstrip open end is taken into account as indicated. The 50- $\Omega$  line width is fixed by 0.6 mm, and all resonators have the same line width of 0.26 mm.

The other dimensions are optimizable, and the initial values are determined as follows:.

The half-wavelength is approximately 12.4 mm; therefore, initially,  $L1 = L2 = L3 = 6.2$  mm, as well as  $L4 + L5 = 6.2$  mm. The tapped location L4 is estimated to be 2.1 mm using the formulation. Once the sweep analysis was complete, we collected the simulation results for each parameter value.

This could include parameters like S-parameters, radiation patterns, impedance, or any other relevant data. I saved the data for further analysis.

After defining the simulation configuration, HFSS conducted electromagnetic simulations to solve Maxwell's equations and obtain the required results. Scattering parameters (S-parameters), electromagnetic field distributions, and other relevant characteristics were included in these studies.

We took the pertinent data out of the HFSS results once the simulations were finished. In order to do this, we can export the simulation data, such as S-parameter matrices or field distributions, and then plot the results into Real and Imaginary parts using one of the following file formats: Touchstone (.snp) files, CSV files, or another format that works with our preferred deep learning framework or tool.



## Chapter III : Experimental Results

	A	B	C	D	E	F	G	H	I	J	K
1	Freq [GHz], "im(S(1,1)) [] - L1='6.418mm' L2='5.936mm' s1='0.283mm' s2='0.404mm'", "im(S(1,1)) [] - L1='6.4185mm' L2='5.936mm' s1='0.283mm' s2='0.404mm' s1='0.283mm' s2='0.405mm'", "im(S(1,1)) [] - L1='6.418mm' L2='5.93675mm' s1='0.284mm' s2='0.405mm'", "im(S(1,1)) [] - L1='6.4185mm' L2='5.93675mm' s1='0.284mm' s2='0.405mm'", "re(S(1,1)) [] - L1='6.418mm' L2='5.9375mm' s1='0.284mm' s2='0.4045mm'", "re(S(1,1)) [] - L1='6.4185mm' L2='5.9375mm' s1='0.284mm' s2='0.4045mm'", "im(S(1,2)) [] - L1='6.418mm' L2='5.93825mm' s1='0.283mm' s2='0.406mm'", "im(S(1,2)) [] - L1='6.4185mm' L2='5.93825mm' s1='0.283mm' s2='0.406mm'", "im(S(1,2)) [] - L1='6.419mm' L2='5.93825mm' s1='0.285mm' s2='0.405mm'", "im(S(1,2)) [] - L1='6.418mm' L2='5.939mm' s1='0.285mm' s2='0.405mm'", "im(S(1,2)) [] - L1='6.4185mm' L2='5.939mm' s1='0.285mm' s2='0.4045mm'", "re(S(1,2)) [] - L1='6.418mm' L2='5.936mm' s1='0.2845mm' s2='0.4045mm'", "re(S(1,2)) [] - L1='6.4185mm' L2='5.936mm' s1='0.2845mm' s2='0.4045mm'", "re(S(1,2)) [] - L1='6.418mm' L2='5.936mm' s1='0.2845mm' s2='0.4045mm'", "re(S(1,2)) [] - L1='6.4185mm' L2='5.936mm' s1='0.2845mm' s2='0.4045mm'"										
2	3.04347826086957, -0.926881054009197, -0.945228808375984, -0.945217117082844, -0.945205428120992, -0.944633890678972, -0.931078708187112, -0.108916950255, 0.0056070036110344, 0.00559463995114059, 0.0055724332499596, 0.00582575497552931, 0.00582787982539644, 0.00575491199080933										
3	3.08695652173913, -0.964728091182337, -0.975064953380685, -0.975059592236563, -0.975054231395624, -0.974798689580955, -0.967623100446545, -0.4094, 0.00515510841963834, 0.00504042506493871, 0.00556221397076994, 0.00506019425705698, 0.00511849501563259, 0.00530222853528529, 0.0049										
4	3.1304347826087, -0.982710519129683, -0.979473407260189, -0.979473360338279, -0.97947331000674, -0.979510147550087, -0.982528322153096, -0.98										
5	8902237, 0.00332419377964673, 0.00439468079014692, 0.00332550818262962, 0.00439270241682124, 0.00439441574568971, 0.00439453437778636, 0.0										
6	3.17391304347826, -0.977014156774996, -0.952817284925492, -0.95282021043881, -0.952823125805672, -0.95309169121033, -0.970963625126019, -0.9										
7	01, 0.00111933556420473, 0.00111354431997019, 0.00111587817563832, 0.00111661036162589, 0.00111550338924694, -0.00286930943817821, 0.00111										
8	3.21739130434783, -0.943016105586336, -0.888220562291285, -0.88822004729492, -0.888223424976968, -0.88861448062571, -0.926831340029752, -0.9										
9	52931, -0.00409598879121058, -0.00551349436029972, -0.00550540272679793, -0.00550564417267348, -0.00552113509050413, -0.0055115401721795, -0										
10	3.26086956521739, -0.874966837420088, -0.777214298472451, -0.777206422616297, -0.777198503350466, -0.777543014761536, -0.842271659238923, -0										
11	0.022266274285292, -0.0222433151565072, -0.0217333942477091, -0.0160440694603376, -0.0214872850918726, -0.0134891356531824, -0.02378933984										
12	3.30434782608696, -0.765647263304909, -0.609636276218638, -0.609606115036154, -0.609575874049441, -0.609641453220677, -0.707160673223518, -0										
13	46306558, -0.0340889781781805, -0.0507303568956924, -0.0506668827560833, -0.0415822570558135, -0.0507689385510923, -0.0415979240996523, -0.0										
14	3.34782608695652, -0.606561679626759, -0.375117111577854, -0.375045768651281, -0.374974288808805, -0.374493487153401, -0.509390464162186, -0										

Figure III.5: Data before preprocessing

The extracted data may need to be cleaned up and formatted appropriately for deep learning, which may involve preprocessing operations. Depending on the particular needs of our deep learning model, this may entail addressing missing data, scaling the data, standardizing values, or doing any other required data transformations.

Freq	L1	L2	S1	S2	re(S11)	im(S11)	re(S12)	im(S12)
3.0	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.5578900805	-0.815274258	0.0106834351	0.0061606666
3.0434782608	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.4487900343	-0.879225778	0.0143939205	0.0060768820
3.0869565217	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.3257514963	-0.930697153	0.0193406864	0.0053153290
3.1304347826	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.1883015995	-0.966320075	0.0259231558	0.0033262888
3.1739130434	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.0361387230	-0.981574141	0.0346227889	-0.0008491523
3.2173913043	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.130534653	-0.970184439	0.0459234481	-0.0088962605
3.2608695652	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.310222208	-0.923182743	0.0600147866	-0.0237893398
3.3043478260	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.498253273	-0.827543025	0.0758769558	-0.0507303568
3.3478260869	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.682290873	-0.664725200	0.0886704939	-0.0983836889
3.3913043478	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.832008396	-0.411757800	0.0828980900	-0.1787823323
3.4347826086	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.879907272	-0.056661239	0.0182436788	-0.2972909422
3.4782608695	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.715408389	0.3408713652	-0.176740458	-0.4073461622
3.5217391304	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.305318982	0.5697399738	-0.511724525	-0.3438361448
3.5652173913	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.1034992855	0.4700881835	-0.757895145	0.0177428957
3.6086956521	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.2875073868	0.2328187897	-0.684796323	0.4805655426
3.6521739130	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.3303127695	0.0464562170	-0.356863273	0.7917250924
3.6956521739	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.3348051056	-0.095110040	0.055329967	0.8733192391
3.7391304347	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.3070663920	-0.226241375	0.4239815295	0.7600516506
3.7826086956	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.2303327632	-0.335737039	0.6888132272	0.5239169000
3.8260869565	6.419mm	5.9375mm	0.2845mm	0.4055mm	0.1136252364	-0.391845144	0.8398771255	0.2249429221
3.8695652173	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.007800015	-0.372635201	0.8810312205	-0.1048151152
3.9130434782	6.419mm	5.9375mm	0.2845mm	0.4055mm	-0.083104820	-0.280640267	0.8017643262	-0.4447840703

Figure III.6: Data after preprocessing

These procedures enable us to use HFSS to produce datasets including the simulation data required for deep learning model training and evaluation for microwave modeling and

design. By feeding these datasets into the deep learning algorithms, the models are able to learn from and anticipate the behavior of the simulated microwaves.

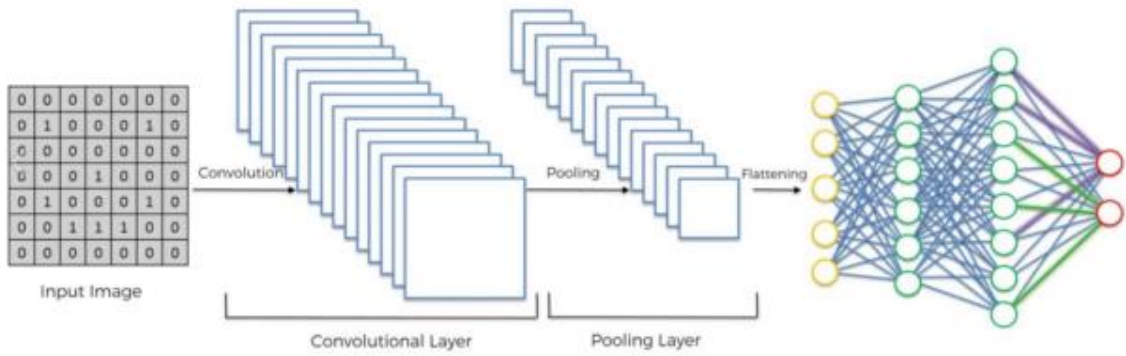
### 3.4. Deep learning model development and architecture

Deep learning model development for microwave modeling and design involves designing and constructing neural network architectures tailored to specific tasks.

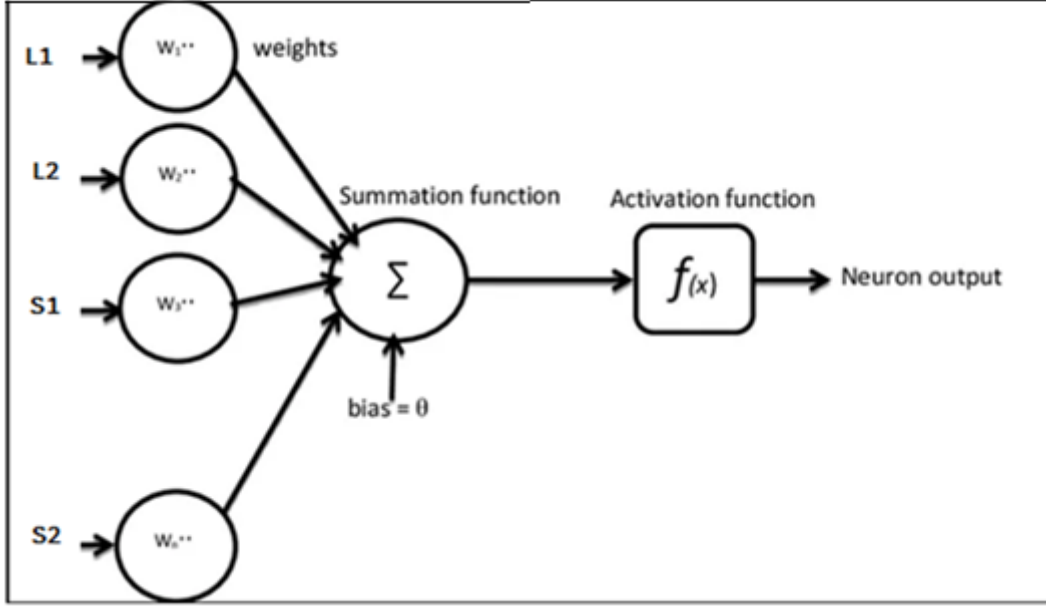
Here's a high-level overview of the process:

First, we defined the problem by specifying the microwave modeling or design task we aimed to address using deep learning. This could involve predicting the performance of microwave components, optimizing system parameters, or automating the design process.

Next, we selected a neural network architecture tailored to our problem and data characteristics. This involved choosing an appropriate type of neural network, such as convolutional neural networks (CNNs) for image-based tasks, recurrent neural networks (RNNs) for sequential data, or a combination of various network types. We considered factors such as the number of layers, activation functions, and connectivity patterns based on the problem's complexity and our computational resources. For our study, we opted for convolutional neural networks (CNNs).



**Figure III.7: Proposed CNN Topology for our study.**



**Figure III.8: Neural Network structure of a single neuron in our Study**

Each neuron in the network has its weights ( $W$ ) and biases ( $b$ ) randomly initialized before the input data is sent through the layers to determine the final output.

Each of the four neurons that make up the input layer represents one input feature. Each neuron multiplies the associated input value by its weight before adding the bias term to the total.

$$z1 = (w11 * L1) + (w21 * L2) + (w31 * S1) + (w41 * S2) + b1 \quad (3.1)$$

The input characteristics in this case are represented by  $L1$ ,  $L2$ ,  $S1$ , and  $S2$ , and the weights connecting the inputs to the first neuron are  $w11$ ,  $w21$ ,  $w31$ , and  $w41$ , the bias term for the initial neuron is  $b1$ .

Next, the weighted sum is subjected to the ReLU activation function, which maintains positive values constant while setting negative values to 0.

$$a1 = ReLU(z1) \quad (3.2)$$

In a similar manner, each neuron in each buried layer goes through the aforementioned stages once more, moving the inputs ahead through the network.

The network determines the loss or mistake by comparing the expected and intended outputs. A suitable loss function, like cross-entropy loss or mean squared error (MSE), can be used for this.



Using the chain rule of calculus, the network determines the loss gradients with respect to the weights and biases. The network uses these gradients to update the weights and biases in the opposite direction of the gradient, thereby reducing the overall loss. This process is known as backpropagation. To regulate the step size of the updates, the weights and biases are changed using an optimization process, such as stochastic gradient descent (SGD), multiplied by a learning rate.

For a predetermined number of epochs or until the network converges to a satisfactory degree of accuracy, the forward propagation, loss calculation, and backpropagation steps are repeated iteratively. During each iteration, the network is able to modify its weights and biases, enhancing its performance on the training set.

### 3.5. Development environment

To model and design microwaves using deep learning, we selected multiple development environments. Here are a few of the alternatives we have used:

- ❖ **Google Colab:** This cloud-based development environment offers GPU and TPU access for free. With an interface akin to a Jupyter Notebook, it enables us to write and run Python code. Colab's seamless Google Drive integration makes data storage and sharing easy.
- ❖ **Google Drive** is a cloud-based solution for file sync and storage offered by Google. It enables users to share, view, and store data on any internet-connected device. While it can be used in conjunction with other development environments, such as Google Colab or local development setups, Google Drive is not a development environment in and of itself for deep learning work.

#### Libraries:

- **TensorFlow:** Google created the open-source deep learning library TensorFlow. It offers an adaptable platform for deep neural network model construction and training. A vast array of tools and modules are available in TensorFlow for activities including building models, evaluating models, and preparing data.
- **Pandas:** is a flexible library for analysis and data manipulation. It offers tabular data structures, like data frames, that are helpful for handling and organizing the data. Prior to deep learning model training, Pandas is frequently used for loading, preprocessing, and exploring datasets.

- **NumPy:** NumPy is a core Python scientific computing package. Among its many features are strong array operations and linear algebraic functions; these are necessary for preparing and modifying the numerical data that deep learning uses.
- **Keras:** Keras is a high-level deep learning library that runs on top of TensorFlow. Running atop TensorFlow is the high-level deep learning library known as Keras. By offering intuitive APIs and abstractions, it makes the process of creating neural networks easier. With Keras, we can quickly prototype models and work with different neural network designs.
- **Matplotlib and Seaborn:** These two data visualization libraries are available online. With their help, we may generate a variety of plots and visualizations, such as line, scatter, histogram, and heatmap plots, to examine and display our data.

### 3.6. Model Training:

When training a CNN model, the training dataset is repeatedly presented to the model, and the loss is computed. The steps needed to train a CNN model for microstrip coupled line design are summarized here.

- **Define the training process:** We gave the model a set number of epochs, or iterations, to train for. The term "epoch" describes a full run through the training dataset.
- **Define the loss function:** We selected a suitable loss function to quantify the difference between the true outputs in the training dataset and the predicted outputs of the model. The type of data and the particular modeling or design task will determine which loss function is best. Custom-defined loss functions, categorical cross-entropy, and mean squared error (MSE) are a few examples of common loss functions.
- **Select an optimizer:** We selected an optimization algorithm that will use the computed loss to update the model's parameters. Stochastic gradient descent is a well-liked optimization algorithm (SGD). The model's training performance may be impacted by the update rules and hyperparameters unique to each optimizer.
- **Training loop:** We went through the training dataset iteratively for the predetermined number of epochs. Give the model the input samples for each epoch to get the anticipated outputs. Utilizing the selected loss function, compute the loss by comparing the expected and actual outputs.

## Chapter III : Experimental Results

---

- **Backpropagation and parameter update:** To determine the gradients of the loss in relation to the model's parameters, we used backpropagation. The gradients indicate which way to update the parameters and how much to change in order to minimize the loss. The optimizer then uses the gradients and the selected optimization algorithm to update the model's parameters appropriately.
- **Monitor training progress:** As the training set is being used, keep an eye on and record metrics like accuracy, loss, and other pertinent evaluation metrics. We were able to evaluate the model's effectiveness and identify any possible problems, like overfitting or under fitting, as a result.
- **Validation set evaluation:** In order to gauge the model's capacity for generalization and avoid overfitting, we routinely assessed its performance on a different validation dataset. On the validation set, we computed evaluation metrics, which we then contrasted with the training metrics. Overfitting may need modifying the model or training procedure if the model's performance on the validation set declines while training metrics keep getting better.
- **Hyperparameter tuning:** The learning rate, batch size, regularization strategies, and network architecture were some of the hyperparameter values we experimented with to determine the optimal combination that maximizes the model's performance on the validation set.
- **Iterate and refine:** We iterated and refined the model, adding regularization techniques, adjusting hyperparameters, and changing the network architecture as needed based on the evaluation results. Until the model performs satisfactorily on both the training and validation sets, we repeated the training procedure.
- **Test set evaluation:** Using a different test dataset that was not used for training or validation, we evaluated the model's performance.

We began the model training process with a Python simulation script because it offers an adaptable and strong environment for fully connected CNN model implementation, training, and evaluation.

Using our Python simulation script, the general process of the suggested deep neural network parametric model is represented as follows:

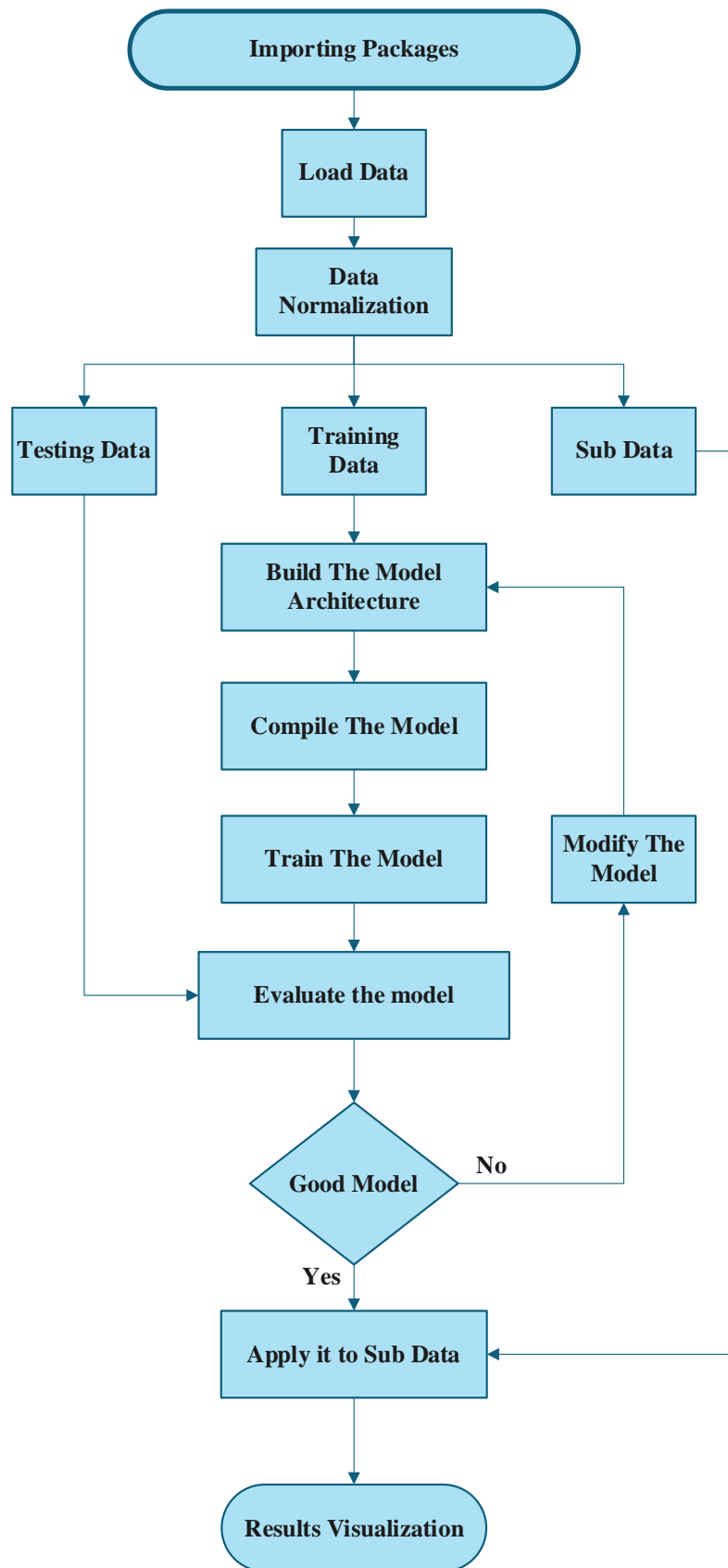


Figure III.9: schematic of the proposed deep neural network parametric model's overall development process

## Chapter III : Experimental Results

---

Importing a few libraries will be our first step; the rest will be added as and when the programs need them at various points. To begin with, we will import the libraries that will enable us to import and get ready the dataset for the model's training and testing.

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
```

Importing the required libraries for our project was our first step. The code snippet imports different libraries into the Python program using a series of import statements. Among these libraires are:

- ✓ **Pandas** for data manipulation and analysis.
- ✓ **Numpy** for numerical computations and array operations.
- ✓ **Tensor flow** for building and training machine learning models.
- ✓ **Sklearn.model\_selection** for splitting data into training and testing subsets.
- ✓ **Sklearn.preprocessing** for data preprocessing tasks such as feature scaling.
- ✓ **Matplotlib.pyplot** for creating visualizations and plots.

```
# Charger les données
data = pd.read_csv('/content/drive/MyDrive/DT/processed_results_re_im (1).csv', sep=';')
data.head()
```

Google Colaboratory (Colab) uses the code snippet from google.colab import drive and drive.mount('/content/drive') to mount and access our Google Drive inside the Colab environment. It enables us to use our Colab notebook to directly access and work with files stored on our Google Drive.

## Chapter III : Experimental Results

The code `data = pd.read_csv('drive/MyDrive/DT/Data.csv', sep=';')` reads a CSV file named `'Data.csv'` from a specified folder in Google Drive using the pandas library. The `pd.read_csv` function loads the CSV file, and the resulting content is stored in a pandas DataFrame called `data`. The parameter `sep=';'` indicates that the file uses a semicolon (;) as the delimiter between values.

the command “data” can used to display the data we have :

	Freq	L1	L2	S1	S2	re(S11)	im(S11)	re(S12)	im(S12)
0	3.000000	6.419	5.9375	0.2845	0.4055	0.557890	-0.815274	0.010683	0.006161
1	3.043478	6.419	5.9375	0.2845	0.4055	0.448790	-0.879226	0.014394	0.006077
2	3.086957	6.419	5.9375	0.2845	0.4055	0.325751	-0.930697	0.019341	0.005315
3	3.130435	6.419	5.9375	0.2845	0.4055	0.188302	-0.966320	0.025923	0.003326
4	3.173913	6.419	5.9375	0.2845	0.4055	0.036139	-0.981574	0.034623	-0.000849
...	...	...	...	...	...	...	...	...	...
43745	5.826087	6.418	5.9390	0.2845	0.4060	0.104480	0.946653	0.000307	-0.000023
43746	5.869565	6.418	5.9390	0.2845	0.4060	0.217566	0.927171	0.000293	-0.000040
43747	5.913043	6.418	5.9390	0.2845	0.4060	0.327385	0.894416	0.000273	-0.000050
43748	5.956522	6.418	5.9390	0.2845	0.4060	0.432585	0.848810	0.000252	-0.000053
43749	6.000000	6.418	5.9390	0.2845	0.4060	0.531801	0.790886	0.000233	-0.000051

43750 rows × 9 columns

- **subdata** = This line creates a new DataFrame called **subdata** by extracting rows from the **data** DataFrame. It effectively selects a subset of the original data for later testing or analysis.
- **data.drop**:. The **drop** function is used to drop specific rows based on their indices. The **inplace=True** parameter ensures that the changes are made directly to the **data** DataFrame itself.
- **features = data[['L1', 'L2', 'S1', 'S2','freq']]**: This line creates a new DataFrame called **features** by selecting specific columns ('L1', 'L2', 'S1', 'S2','freq') from the **data** DataFrame. The double square brackets **['']** are used to select multiple columns as a subset of the original DataFrame.

## Chapter III : Experimental Results

---

- **labels = data[['Re', 'Im']]:** This line creates a new DataFrame called **labels** by selecting columns 'Re' and 'Im' from the **data** DataFrame. It creates a subset of the original DataFrame containing only these two columns.

```
# Normalisation des caractéristiques
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_features = scaler.fit_transform(features)

# Division des données en ensembles d'entraînement et de test
train_features, test_features, train_labels, test_labels = train_test_split(
    scaled_features, labels, test_size=0.2, random_state=40, shuffle=False)

# Reshape des données pour le CNN
train_features = train_features.reshape(train_features.shape[0], train_features.shape[1], 1)
test_features = test_features.reshape(test_features.shape[0], test_features.shape[1], 1)
```

We have used the `MinMaxScaler` class to implement feature scaling. In order to keep all features on a similar scale and prevent any feature from predominating over the others during model training, this technique is crucial to machine learning.

We assigned the `scaler` variable to an instance of the `MinMaxScaler` class that we had created.

Next, we used the `fit_transform()` function of the `MinMaxScaler` object to apply the scaling transformation to the features DataFrame. Each feature's minimum and maximum values were determined using this method, and the feature values were scaled appropriately. The scaled features that are produced are kept in the `scaled_features` variable and can be utilized for machine learning or additional analysis. We have normalized the feature range through feature scaling, which can aid in enhancing the efficiency and convergence of machine learning algorithms. It prevents bias based on the features' original scales and guarantees that each feature contributes equally.

The data is split into training and testing sets using the `train_test_split` function from `scikit-learn`.

- **train\_features, test\_features, train\_labels, test\_labels = train\_test\_split(scaled\_features, labels, test\_size=):** This line splits the scaled features (`scaled_features`) and labels (`labels`) into training and testing sets. The `train_test_split` function randomly shuffles the data and divides it into sets based on the specified `test_size` parameter. The training set features are stored in



## Chapter III : Experimental Results

**train\_features**, the testing set features in **test\_features**, the training set labels in **train\_labels**, and the testing set labels in **test\_labels**.

We can train a machine learning model on the training set and assess its performance and generalization abilities on the testing set by dividing the data into training and testing sets. This aids in determining how well the model will function with unknown data.

```
# Construction du modèle CNN
model = Sequential()
model.add(Conv1D(64, 2, activation='relu', input_shape=(train_features.shape[1], 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(128, 2, activation='relu'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(4)) # 4 sorties pour les parties réelles et imaginaires

# Compilation du modèle
model.compile(optimizer='adam', loss='mean_squared_error')
```

The neural network architecture is formed by building the sequential model by stacking these layers one on top of the other. Each dense layer applies a linear transformation to its inputs, followed by the specified activation function.

Several hidden layers are included in this architecture's design to help identify intricate relationships and patterns in the data. Real and imaginary values are represented by the four units in the output layer(S11 Re,S11 Im & S21 Re,S21 Im).

We define the loss function to assess the model's performance and the optimization algorithm to be used during training.

- ✓ **optimizer='adam'**: The optimizer used is Adam, a widely adopted optimization algorithm in deep learning. Adam automatically adjusts the learning rate throughout training, enabling faster convergence and better handling of various types of data.
- ✓ **loss='mean\_squared\_error'**: The loss function used is mean squared error (MSE), which is commonly applied in regression tasks. MSE computes the average of the squared differences between the predicted and actual values. By minimizing MSE during training, the model improves its ability to make predictions that are closer to the actual values.

The model is trained using the training features ('train\_features') and training labels ('train\_labels'). The training runs for 150 epochs, where each epoch represents one complete pass through the entire training dataset. The 'validation\_data' parameter is used to assess the model's performance on the given testing features ('test\_features') and testing labels ('test\_labels') during training.



## Chapter III : Experimental Results

```
Epoch 1/150
137/137 ————— 3s 10ms/step - loss: 0.1969 - val_loss: 0.1767
Epoch 2/150
137/137 ————— 3s 12ms/step - loss: 0.1730 - val_loss: 0.1611
Epoch 3/150
137/137 ————— 2s 7ms/step - loss: 0.1596 - val_loss: 0.1550
Epoch 4/150
137/137 ————— 1s 7ms/step - loss: 0.1520 - val_loss: 0.1506
Epoch 5/150
137/137 ————— 1s 7ms/step - loss: 0.1496 - val_loss: 0.1454
Epoch 6/150
137/137 ————— 1s 7ms/step - loss: 0.1439 - val_loss: 0.1412
Epoch 7/150
137/137 ————— 1s 7ms/step - loss: 0.1396 - val_loss: 0.1365
Epoch 8/150
137/137 ————— 1s 7ms/step - loss: 0.1350 - val_loss: 0.1328
Epoch 9/150
137/137 ————— 1s 7ms/step - loss: 0.1311 - val_loss: 0.1266
Epoch 10/150
137/137 ————— 1s 7ms/step - loss: 0.1248 - val_loss: 0.1221
```

**Predictions** = `model.predict(test_features)`: This line uses the trained model to generate predictions based on the provided test features (`test_features`). The `predict` method applies the model to the input data and returns the corresponding output predictions.

```
The Predaction: [ 0.20243488 -0.9055777  0.0500933  0.12562723]
The Real Value: [ 0.17616645 -0.97232381  0.01434669  0.00280281]
```

```
The Predaction: [ 0.07807197 -0.96701425  0.03730608  0.0743991 ]
The Real Value: [ 0.0470046  -0.98615187  0.0183108  0.00119474]
```

```
The Predaction: [-0.04870848 -1.0140736  0.03780938  0.02616154]
The Real Value: [-0.08685445 -0.98221402  0.02329294 -0.00160172]
```

```
The Predaction: [-0.17677416 -1.0592915  0.04243232 -0.0205712 ]
The Real Value: [-0.22394369 -0.95853285  0.02951134 -0.00628526]
```

```
The Predaction: [-0.30790946 -1.0896806  0.04641677 -0.05030648]
The Real Value: [-0.36253202 -0.91257669  0.03715574 -0.01399775]
```

The actual values and the expected values are somehow distant because the model we trained has not been trained enough to achieve better results.

### 3.7. Results and discussion

In the simulation, we used our dataset to train our CNN model. The following are the findings of this analysis:

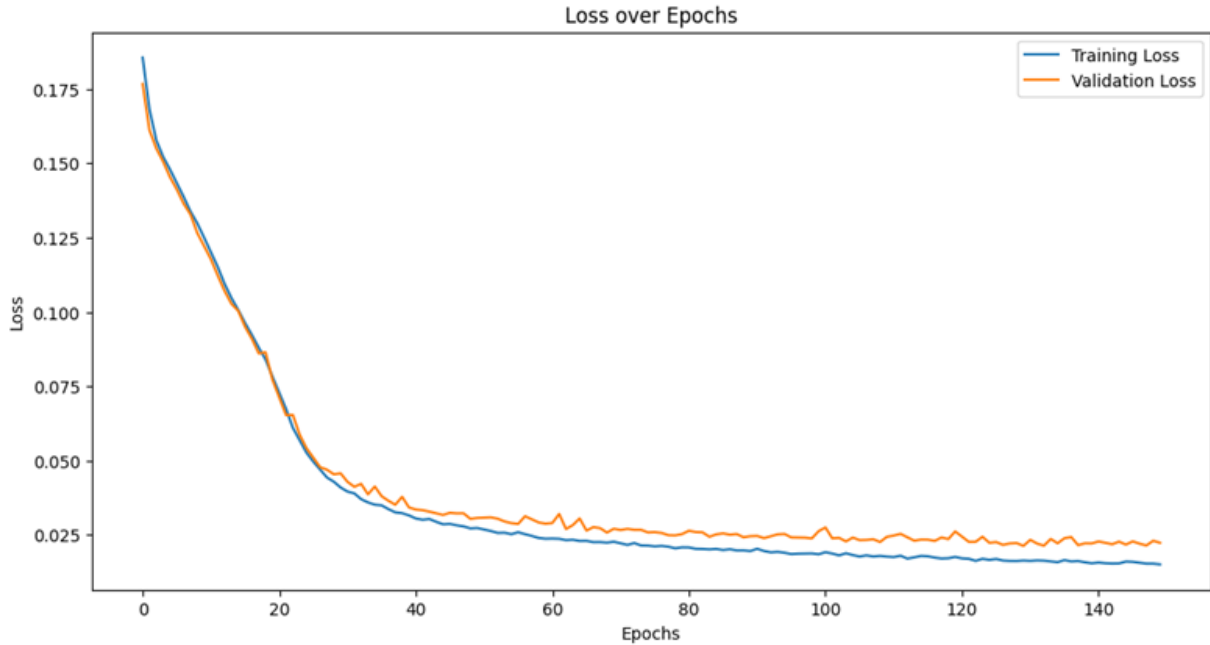


Figure III.10: model's learning progress over the epochs

The graph displays the training and validation loss during the training process of a model.

The model's fit to the training set is indicated by the training loss, which is represented by the blue line. The training loss's declining trend indicates that the model is improving over the epochs and learning new things. This shows that the patterns and structure of the training data are being well captured by the model.

The validation loss, which gauges the model's effectiveness using unseen validation data, is represented by the red line. The model does not overfit to the training set and is effectively generalizing, as evidenced by the decreasing trend of the validation loss. This shows that the model's predictions hold up well and are in line with fresh data.

When both the training and validation losses progressively converge or stabilize, it can be assumed that the model has successfully balanced learning from the training data with generalizing to new data. This indicates that the model is likely doing well and has found the optimal set of parameters in terms of reducing the loss.

### 3. 7.1 Comparison of real part of S11 & Frequency:

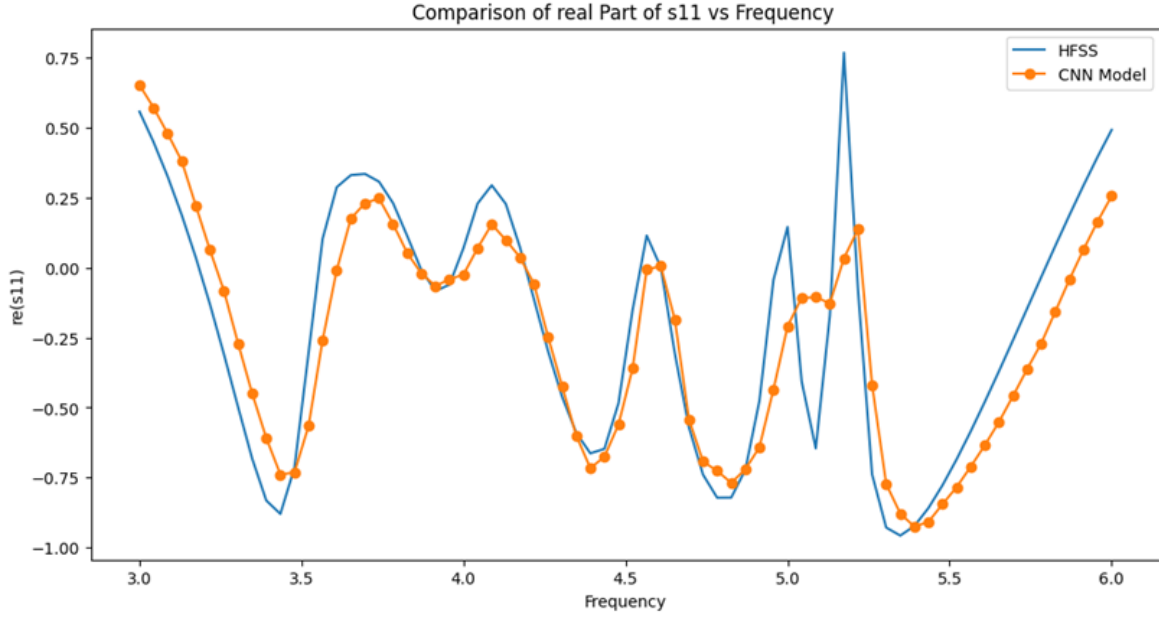


Figure III.11: Plot of S11 real part & frequency

By comparing the blue and orange lines, we can assess the accuracy of the model's predictions. The closer the orange line aligns with the blue line, the more accurate the model's predictions are in capturing the real part of the data S11.

The graph indicates that the model can make precise predictions and has successfully learned the real portion of the data set.

### 3. 7.2 Comparison of imaginary part of S11 & Frequency:

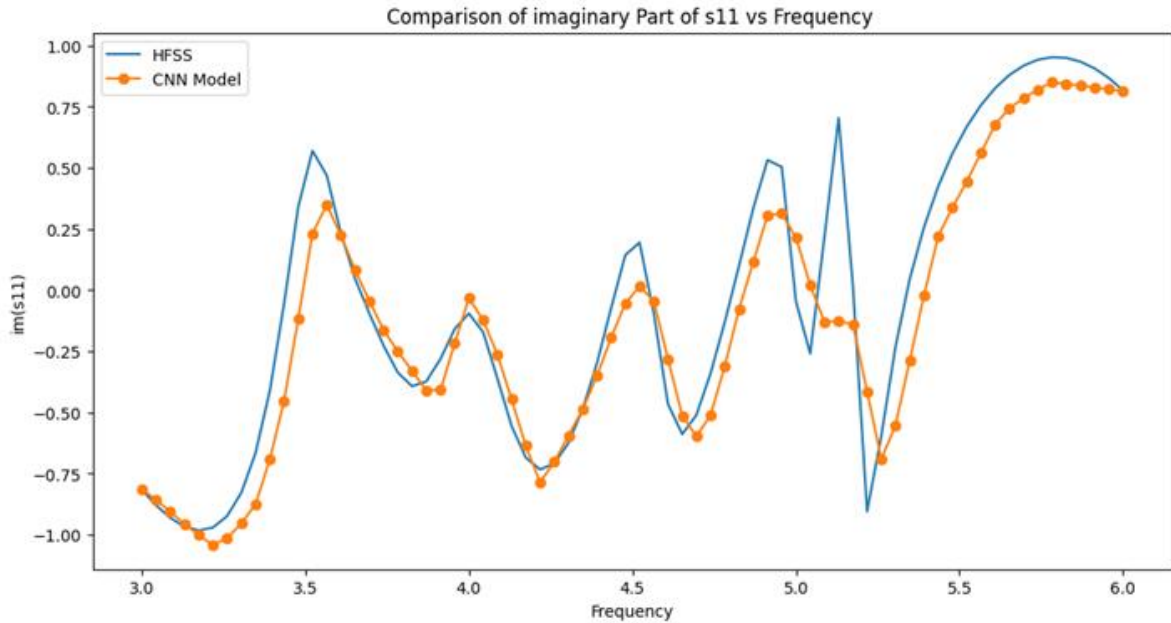


Figure III.12: Plot of S11 imaginary part & frequency

## Chapter III : Experimental Results

By comparing the blue and orange lines, we can assess the accuracy of the model's predictions. The closer the orange line aligns with the blue line, the more accurate the model's predictions are in capturing the imaginary part of the data S11.

The graph indicates that the model can make precise predictions and has successfully learned the imaginary part of the data set

### 3. 7.3 Comparison of real part of S21 & Frequency:

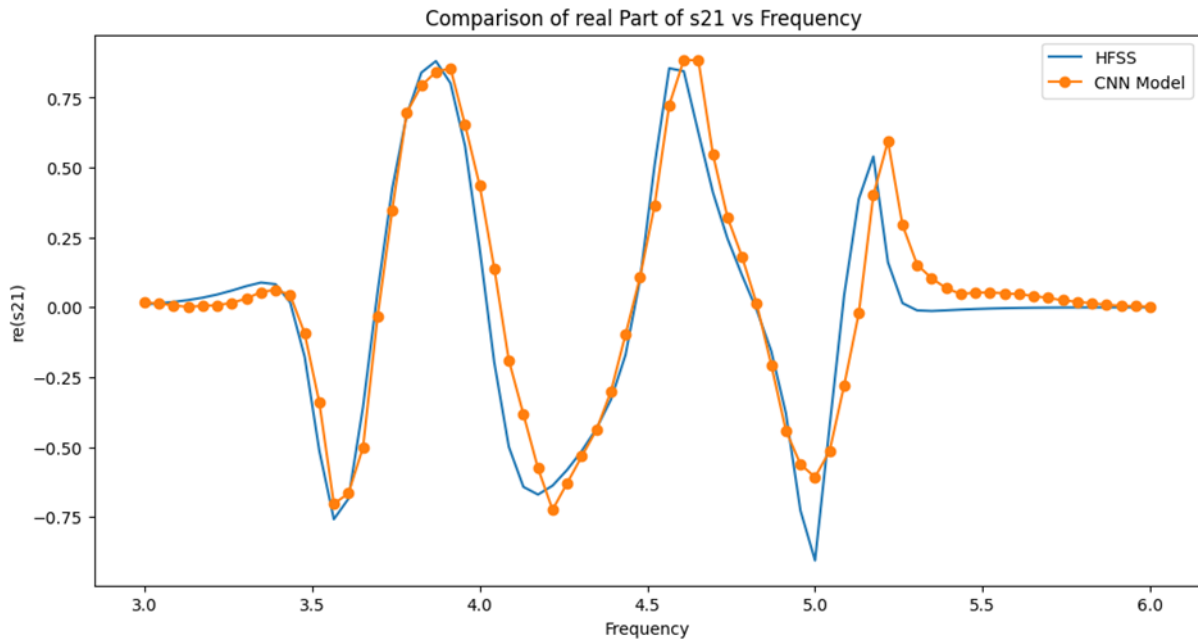
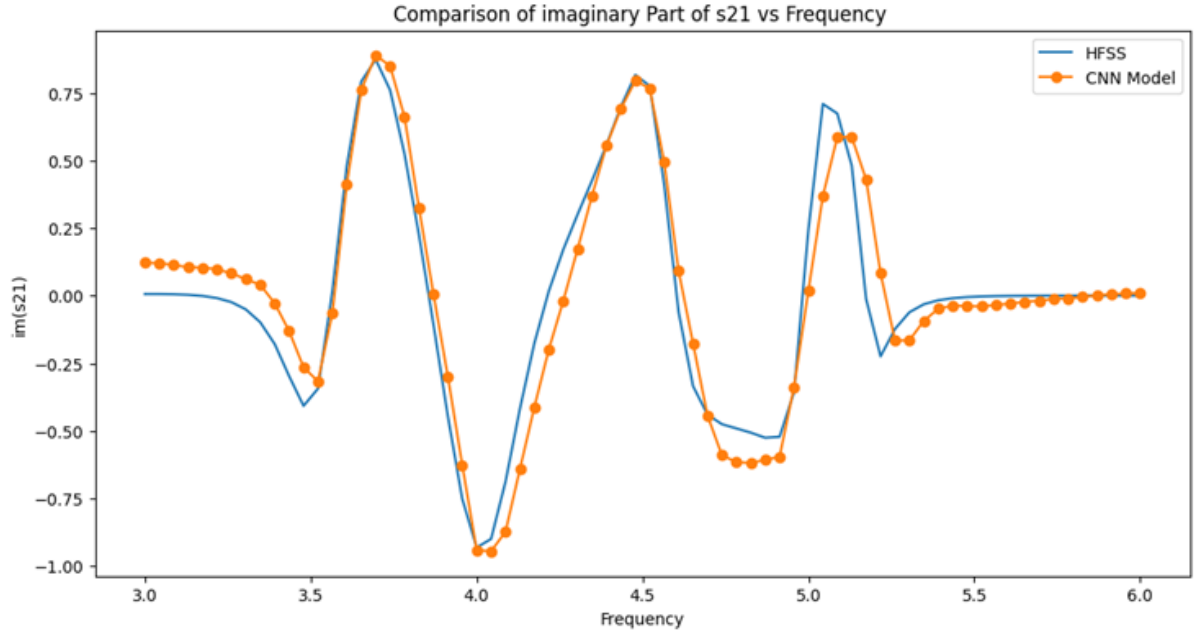


Figure III.13: Plot of S21 real part & frequency

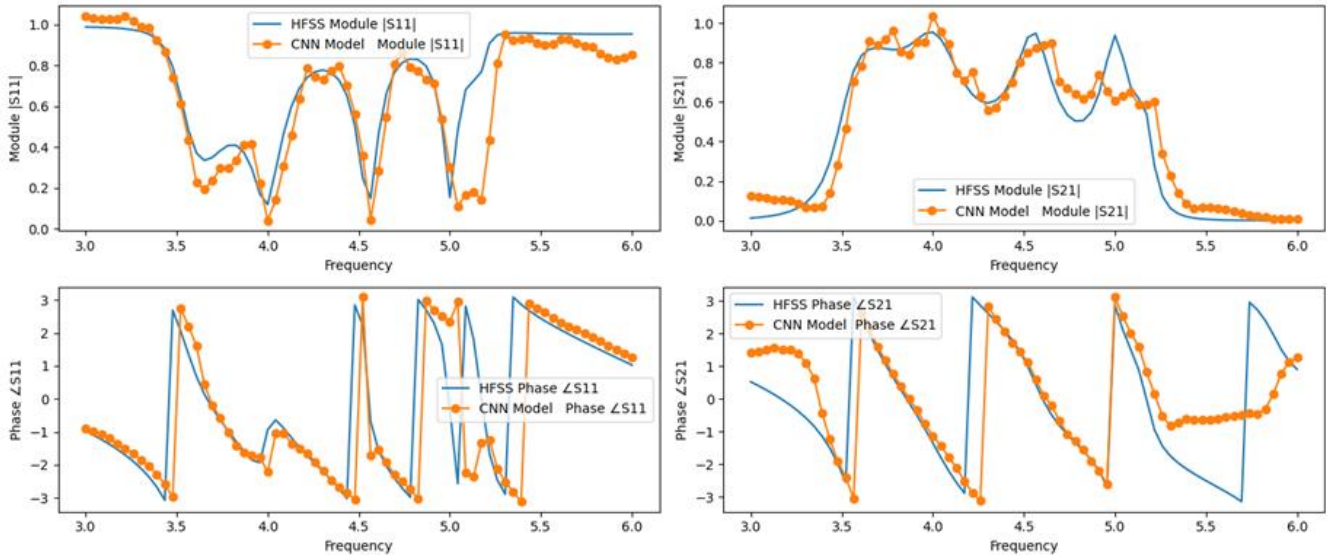
We can evaluate the model's prediction accuracy by contrasting the blue and orange lines. The more closely the orange and blue lines match, the more closely the model's predictions reflect the true portion of the data S21.

### 3. 7.4 Comparison of Imaginary part of S21 & Frequency:



**Figure III.14: Plot of S21 imaginary part & frequency**

The comparison between the blue and orange lines reveals the accuracy of the model's predictions for the imaginary component of the S21 data. When the orange line closely coincides with the blue line, it signifies that the model has effectively learned the underlying pattern and can accurately predict the imaginary part.



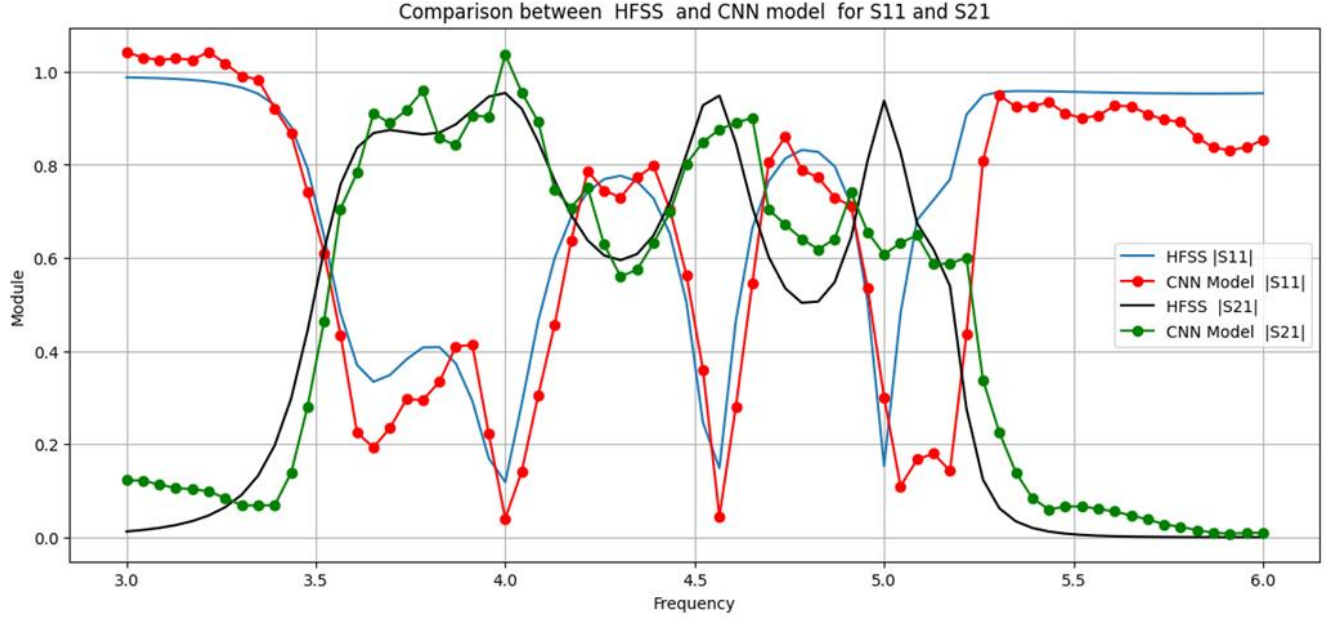
**Figure III.15: Magnitude & phase of S11, S21 plotted as function of frequency**

The graph shows how the magnitude and phase values of S11 and S21 compare to the magnitude and phase values that a model predicts. Since the output results meet the

## Chapter III : Experimental Results

expected magnitude and phase values, we can conclude that the data set's magnitude and phase values are fairly close to the predictions.

### 3.7.5. Comparison between HFSS & CNN model for S11 and S21



**Figure III.16: HFSS & CNN model plotted as function of frequency of S11, S21.**

The CNN and HFSS models of S11 and S21 are contrasted in the graph. Because we only ran 625 simulations, the model did not have enough training data to produce results that were more accurate. Based on this, we can conclude that CNN performed an adequate job of prediction.

### 3.8. Conclusion:

CNN did a good job in training on the data we input, regardless of the estimated small number of simulations, which is 625. Therefore, if we increase the number of simulations, we will certainly achieve better training results.

# Conclusion



## General Conclusion

To sum up, the goal of applying deep learning to microstrip design and analysis is to improve the process's precision, effectiveness, and automation. Conventional techniques for microstrip design and analysis can be labor-intensive and error-prone since they frequently call for manual intervention and specialized knowledge. The goal is to create models that can recognize patterns and relationships from huge datasets by utilizing deep learning techniques. This will allow for more precise predictions and effective microstrip structure optimization.

In general, the goal of incorporating deep learning techniques into microstrip design and analysis is to transform the field by offering automated, more precise, and efficient solutions. Improved microstrip designs in terms of performance, reliability, and manufacturability result from its ability to enable designers to investigate intricate design spaces, optimize performance metrics, and expedite the entire design process.

Three chapters comprise the research methodology, which covers the theory of filters and convolutional neural networks. The experimental part involves the use of deep learning (CNN) in microstrip design.

We installed a variety of microstrip designs in the HFSS environment in order to gather data for our study. We established the microstrip structures' geometric parameters, material composition, and other pertinent details. We conducted electromagnetic simulations using HFSS in order to determine the relevant performance metrics, including bandwidth, radiation patterns, and impedance. We tested a variety of dimensions, substrate characteristics, and other important variables by methodically adjusting the microstrip structures' design parameters during the simulation process. Following the simulations, we took the data out of HFSS.

We investigated different neural network architectures that are appropriate for microstrip design and analysis for deep learning model development and architecture. We considered convolutional neural networks (CNNs) in light of the intended predictions and characteristics of the microstrip data. The number and kinds of layers, activation functions, and connectivity patterns that best captured the complex relationships between the microstrip design parameters and performance metrics were determined when designing the architecture. After which we started training the model. To assess the



model's performance, we split the dataset into training and validation sets. To reduce the prediction errors and iteratively update the model's parameters, we employed optimization algorithms. To ensure that the model was successfully converging, we closely monitored the training process by tracking metrics like loss and accuracy. We tried a variety of hyperparameters. Our goal was to develop a deep learning model for microstrip design that was reliable and accurate by using a methodical approach that included data preparation, deep learning model development, and model training. Our research's ultimate objective was to use deep learning to improve the precision and effectiveness of microstrip design predictions, enabling quicker and more optimized designs for a range of performance metrics like bandwidth, radiation patterns, and impedance.

Our research on applying deep learning to microstrip design has produced very encouraging results, showing notable gains in efficiency and accuracy over conventional techniques. The remarkable performance of the deep learning (CNN) model that was developed validates the efficacy of this approach in the domain of microstrip design. The model's capacity to investigate the design space and produce creative design variations was helpful in coming up with original and efficient solutions. With this exploratory capability, designers can push the limits of microstrip performance with new insights and opportunities.

# Bibliography

## Bibliography

1. S. A. Sadrossadat, Y. Cao, and Q.-J. Zhang, "Parametric modeling of microwave passive components using sensitivity-analysis-based adjoint neural-network technique," *IEEE Trans. Microw. Theory Techn.*, vol. 61,
2. F. Feng, C. Zhang, J. Ma, and Q.-J. Zhang, "Parametric modeling of EM behavior of microwave components using combined neural networks and Pole-Residue-Based transfer functions," *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 1, pp. 60\_77, Jan. 2016.
3. K. L. Melde, H.-J. Park, H.-H. Yeh, B. Fankem, Z. Zhou, and W. R. Eisenstadt, "Software Defined Match Control Circuit Integrated with a Planar Inverted F Antenna," *IEEE Trans. Antennas Propagat.*, Vol. 58, No. 12, pp. 3884–3890, Oct. 2010.
4. N. Leszczynska, I. Couckuyt, T. Dhaene, and M. Mrozowski, "Low-cost surrogate models for microwave filters," *IEEE Microw. Wireless Compon. Lett.*, vol. 26, no. 12, pp. 969\_971, Dec. 2016.
5. D. M. Nashaat, H. A. Elsadek, and H. Ghali, "Single Feed Compact Quad-Band PIFA Antenna for Wireless Communication Applications," *IEEE Trans. Antennas Propagat.*, Vol. 53, No. 8, pp. 2631–2635, Aug. 2005.
6. Q.-J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, "Artificial neural networks for RF and microwave design-from theory to practice," *IEEE Trans. Microw. Theory Techn.*, vol. 51, no. 4, pp. 1339\_1350, Apr. 2003.
7. J. E. Rayas-Sanchez, "EM-based optimization of microwave circuits using artificial neural networks: The state-of-the-art," *IEEE Trans. Microw. Theory Techn.*, vol. 52, no. 1, pp. 420\_435, Jan. 2004.
8. M. B. Steer, J. W. Bandler, and C. M. Snowden, "Computer-aided design

of RF and microwave circuits and systems," IEEE Trans. Microw. Theory Techn., vol. 50, no. 3, pp. 996\_1005, Mar. 2002.

9. C. Soras, M. Karaboikis, G. Tsachtsiris, and V. Makios, "Analysis and Design of an Inverted-F Antenna Printed on a PCMCIA Card for the 2.4 GHz ISM Band," IEEE Antennas Propagat. Magazine, Vol. 44, No. 1, pp. 37–44, Feb. 2002.

10. H. Kabir, L. Zhang, M. Yu, P. Aaen, J. Wood, and Q.-J. Zhang, "Smart Modeling of microwave devices," IEEE Microw. Mag., vol. 11, no. 3, pp. 105\_118, May 2010.

Voici une proposition de référence pour ce paragraphe :

11. R. Garg, I. J. Bahl, and M. Bozzi, \*Microstrip Lines and Slotlines\*, 3rd ed., Artech House, 2013.

12. D. M. Pozar, *Microwave Engineering*, 4th ed., Wiley, 2011.

13. Justin Salamon, Juan Pablo Bello. (Novembre 2016), Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification, IEEE SIGNAL PROCESSING LETTERS.

14. Anne Bonner. (Feb 2, 2019), The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification

16. Purit Punyawiwat, Natchuta Wattanapenpaiboon. (Feb 8, 2018) , Interns Explain CNN

19. Richmond Alake. (aug 14, 2020), Implementing AlexNet CNN Architecture Using TensorFlow 2.0+ and Keras

## Webographie

15. <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>
17. <https://iq.opengenus.org/convolutional-neural-networks/>
18. <http://www.cs.kumamoto-u.ac.jp/epslab/ICinPS/Lecture-2.pdf>
20. <https://www.guru99.com/backpropagation-neural-network.html>
21. <https://iq.opengenus.org/convolutional-neural-networks/>