

**Machine Learning for Error Estimation and Compensation for
Microwave Sensors**

by

Nazli Kazemi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Electromagnetic and Microwave

Department of Electrical and Computer Engineering

University of Alberta

©Nazli Kazemi, 2020

Abstract

Microwave planar resonators have been widely used in material characterization, environmental monitoring, proximity sensing, mechanical motion detection, etc. In general, non-contact sensing, real-time measurement, and CMOS compatibility, makes them interesting for chemical sensing, especially in harsh environments. However, the planar nature leaves them vulnerable to the ambient changes with potential impact on the perception of the material under test. Temperature, as one of the most important factors that affects the sensor response, is considered an undesired environmental impact and relevant methods are introduced to tackle it. Even though many types of human- or environment-centric errors such as material displacement, uncontrolled pressure, and relative humidity could be easily removed from the sensor response using conventional methods; compensation of temperature is complicated enough to look for machine learning algorithms as the main focus of this thesis.

In this study, a split ring resonator (SRR) operating @ 1 GHz is designed to measure methanol/acetone content level in water. The planar sensor holding the material inside a tube and a commercial temperature sensor are located inside a chamber. LabView is employed to record scattering parameters during a temperature cycle from room temperature up to 50 °C.

Simulation results suggest the dependency of both material- and sensor-properties on the temperature. It is also shown that an uncontrolled environmental situation can deviate the sensor response into incorrect or meaningless results, based on which our decision is made in industrial application. Transmission/Reflection response of the sensor with all features including resonance frequency, magnitude, and quality factor are examined with the aim of reliable network input data.

Various machine learning algorithms, including Decision Tree, KNN, Logistic Regression, and MLP are examined to remove the effect of temperature on sensor response, whose performance comparison for classifying 10 materials concluded MLP as the best classifier with 100 % accuracy.

But for all that, eliciting environmental status, besides recognizing the material type, is lucrative information to be utilized in further data-processing. Temperature of the measurement, imbued in the recorded transmission profiles, is extracted using a novel technique of cascading a primary classifier with linear regression.

Finally, in pursuit of improving the limit of detection in the microwave sensor, MLP algorithm is significantly scrutinized with the help of hyperparameter optimization, wherein concentrations of methanol-in-water are discriminated with increments as low as 1 % with accuracy of >95%.

Preface

This thesis presents an artificial intelligence-based method to enhance the robustness of planar microwave sensors that was an outcome of a constructive collaboration between Prof. Musilek and Prof. Daneshmand's group. In all chapters presented in this thesis, Nazli Kazemi is the main student contributor.

In chapter 3, the design and fabrication of the microwave sensor used for error compensation involves contribution and help of Mohammad Abdolrazzaghi, who holds research assistantship of Prof. Daneshmand's group as well as Electromagnetic Lead of Phase Advanced Sensors Corp.

Dedication

To my *lovely* **parents**,
without whom,
this amount of hope for life,
love of work,
and persistence through the hardships
would never be possible.

Acknowledgements

Dr. Mojgan Daneshmand, who opened the gate way to my career and support me more than I deserve it during my graduate study at the University of Alberta. Maybe it is one of the hardest times in my life that I want to write about her after seven months that I lost her support, her beautiful smile, and her kindness. If I have the opportunity to study at the university of Alberta and so many others that they are countless, it is because of her hardworking during her life to became one of the professors at this university . I would thank wholeheartedly from Mojgan, my best friend who affect more than anyone on my life because of her relentless care for my progress and development.

THANK YOU SO MUCH MOJGAN FOR EVERYTHING.

I wish to express my sincere appreciation to my current supervisor, Professor Petr Musilek, who has provided me with his support, guidance and patience throughout my studies under his supervision. Without his persistent help, the goal of this project would not have been realized. He graciously undertook my responsibility after Mojgan. Amidst all the turmoil and confusion, I went through after the shocking news I received in January 2020, he was the one who encouraged me to pull together and become strong enough to resume my studies. He is directing assistance towards my career turned out to be more than a supervisory role.

I would like to extend my gratitude to sincerely respectful Dr. Marek Reformat for his generous support on providing me the opportunity to continue my degree along with many other supports during my study.

My sincere appreciation goes to my dear mentor in M2M group, Mohammad Abdolrazzaghi who helped me since I started my journey in the group till the end. To shape my research, I was privileged of his deep discussions and insightful suggestions on various topics of microwave sensors. He happens to be one of the most knowledgeable and creative scholars in his field of study who helped me get through my degree. His being lucratively workaholic is quite impressive.

The physical and technical contribution of 'Future Energy System' is truly appreciated. Without their support and funding, this project could not have reached its goal.

I would like to recognize the invaluable assistance that the staff at the Electrical and Computer Engineering Department of University of Alberta provided during my study, specially Pinder and Wendy who are undeniable part of my accommodation at the department.

Table of Contents

Abstract	ii
Preface	iv
Dedication	v
Acknowledgements	vi
Table of Contents	ix
List of Tables	xii
List of Figures	xiii
List of acronyms	xvi
List of symbols	xviii
CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION	1
1.2 OBJECTIVE	3
1.3 THESIS OUTLINE.....	4
CHAPTER 2 LITERATURE REVIEW AND BACKGROUND	6
2.1 DUMBBELL-SHAPED DEFECT GROUND STRUCTURE.....	7
2.2 TRANSMISSION LINES TERMINATED WITH <i>LC</i> RESONATORS	8
2.3 RAT-RACE BASED DIFFERENTIAL SENSOR	11
2.4 SRR-BASED RELATIVE HUMIDITY COMPENSATION SCHEME	13
2.5 DUAL-MODE SRR TO ELIMINATE RH	15
2.6 DUAL MODE MICROWAVE MICROFLUIDIC SENSOR	17
2.7 SYMMETRIC CPW SENSOR WITH IDC FOR PERMITTIVITY CHARACTERIZATION	20
2.8 MATERIAL IDENTIFICATION USING CSRR ARRAY AND MACHINE LEARNING	22
2.9 MICROSTRIP COMPLEMENTARY SPLIT-RING RESONATOR (MCSRR)	24
CHAPTER 3 MICROWAVE SENSOR DESIGN	28
3.1 SCATTERING PARAMETERS OF THE SENSOR	28
3.2 PROBLEM STATEMENT	32
3.3 MUT PREPARATION	38
CHAPTER 4 DATA PREPARATION	44

4.1	DATA SET PREPARATION	44
4.1.1	<i>Data as .s2p file</i>	44
4.1.2	<i>Extracting S₂₁ as dataset</i>	45
4.1.3	<i>Making a Mega-Matrix containing all the classes</i>	46
4.1.4	<i>Feature Extraction</i>	47
4.2	INPUT OF THE NETWORK	47
4.2.1	<i>Decent S₂₁ Profile</i>	48
4.2.2	<i>Connection between input and output</i>	52
4.2.3	<i>Shuffling the Dataset</i>	52
4.2.4	<i>Imbalanced data</i>	53
4.2.5	<i>Standard Scalar</i>	54
4.2.6	<i>Assigning X and y data</i>	55
4.3	CONCLUSION	55
CHAPTER 5 MACHINE LEARNING ALGORITHMS TO CHARACTERIZE MATERIALS WITH VARYING TEMPERATURE.....		57
5.1	CLASSIFICATION ALGORITHMS	58
5.1.1	<i>KNN</i>	58
5.1.2	<i>Decision Tree:</i>	60
5.1.3	<i>MLP:</i>	61
5.1.4	<i>Logistic Regression & PCA</i>	66
5.1.5	<i>Comparative analysis</i>	69
5.2	LINEAR REGRESSION ALGORITHM	69
5.3	MATERIAL CLASSIFICATION FOLLOWED BY TEMPERATURE REPORTING.....	75
5.4	CONCLUSION	79
CHAPTER 6 HYPERPARAMETER OPTIMIZATION.....		81
6.1	HYPERPARAMETER OPTIMIZATION WITH KERAS	83
6.2	HOW TO TUNE THE NUMBER OF NEURONS IN THE HIDDEN LAYER	86
6.3	HOW TO TUNE BATCH SIZE AND NUMBER OF EPOCHS	88
6.4	HOW TO TUNE KERNEL INITIALIZER	90
6.5	HOW TO TUNE THE TRAINING OPTIMIZATION ALGORITHM.....	92
6.5.1	<i>Stochastic gradient descent</i>	93
6.5.2	<i>Adadelta</i>	94
6.5.3	<i>RMSProp Gradient Descent</i>	95
6.5.4	<i>Adam— Adaptive Moment Estimation</i>	96
6.5.5	<i>Nadam - Nesterov-accelerated Adaptive Moment Estimation</i>	97

6.6	HOW TO CHOOSE LOSS FUNCTIONS WHEN TRAINING DEEP LEARNING NEURAL NETWORKS.....	99
6.6.1	<i>Mean Squared Error Loss</i>	100
6.6.2	<i>Binary Cross-Entropy Loss</i>	100
6.6.3	<i>Multi-Class Cross-Entropy Loss</i>	101
6.6.4	<i>Kullback Leibler Divergence Loss</i>	101
6.7	HOW TO TUNE THE NEURAL ACTIVATION FUNCTION	104
6.8	CONCLUSION	109
CHAPTER 7	CONCLUSIONS AND FUTURE WORK.....	110
7.1	THESIS CONCLUSIONS AND CONTRIBUTIONS.....	110
7.2	SUGGESTIONS FOR FUTURE WORK	111
References	112

List of Tables

Table 3.1 Dielectric constant values for various materials at different temperatures 35

Table 5.1 Comparison between different algorithms..... 69

Table 6.1 Activation function definitions 106

List of Figures

Figure 2.1 (a) Dumbbell-shaped resonator design with two sensing regions for measurement and reference sensors, (b) Microfluidic channels mounted on the planar resonator 8

Figure 2.2 (a) Schematic of transmission line terminated with LC resonator, (b) electrical circuit equivalence of the resonator, (c) Frequency changes in the sensing resonator with sensitivities... 10

Figure 2.3 (a) Schematic of rat-race based reflective-mode differential sensor, (b) Measurement scenario, (c) Output of the sensor in discriminating various concentrations of IPA in DI water. 12

Figure 2.4 (a) Schematic of double loss-compensated SRR sensor covered with sand, (b) Both reference and sensing resonator are impacted by the RH, (c) Sensor is covered with sand experimentally and methanol/ethanol/water are calibrated to recover regardless of the applied RH. 14

Figure 2.5 (a) Dual-mode microwave planar sensor, (b) variation in sensor response when various MUT are injected into sensor with varying RH, (c) Sensor response to permittivity range of 1-80 without RH 16

Figure 2.6 Dual-mode resonator design, (b) Chloroform corrected resonant frequency and (c) quality factor with respect to temperature 18

Figure 2.7 (a) Real and (b) imaginary part of the permittivity of water when measured and corrected with respect to temperature change 19

Figure 2.8 (a) Photo of the fabricated sensor, (b) Experimental setup, (c) Artificial neural network structure for permittivity extraction, (Excerpt from ***) 20

Figure 2.9 The measured frequency shift and amplitude value change for various materials (Excerpt from ***)..... 21

Figure 2.10 (a) Array of CSRRs with frequencies of 1.36 GHz, 3.09 GHz, 5 GHz, 6.82 GHz, and 8.91 GHz, (b) Resonance frequencies of the array 22

Figure 2.11 Total classification accuracy versus combination index for classifying cardboard using (a) LOO- and (b) SKF-cross-validation..... 23

Figure 2.12 Microstrip Complementary Split-Ring Resonator loaded with microfluidic channels, (b) Variation in the first resonator when the material inside varies, (c) Stability of reference resonator regardless of the material variation on sensing resonator 25

Figure 2.13 Structure diagram of the BP-NN, [Excerpt from ***) 26

Figure 2.14 Schematic of the sensor exposed to MUT in varying temperature..... 27

Figure 3.1 Schematic of SRR with dimensions all in [mm]. 28

Figure 3.2 Schematic of microwave sensor and measurement setup..... 29

Figure 3.3 S-parameters of the sensor 31

Figure 3.4 Dielectric spectrum of (a) Acetone, (b) Methanol, (c) Water at various temperatures based on single Debye model 34

Figure 3.5 Flow graph for generation of dielectric constant values for various concentrations on materials in water at different temperatures	36
Figure 3.6 Simulating all materials inside tubing of sensor at various temperatures in HFSS, the inset focuses on a narrow bandwidth to demonstrate the complexity in selecting the correct graph with respect to the resonance frequency	37
Figure 3.7 Repeatable and reproduceable temperature cycle in frequency of operation and its coincidence with the temperature	38
Figure 3.8 Frequency of resonance change in (a) acetone-water and (b) methanol-water solutions with volumetric ratio of 0 : 20% : 100 % over temperature cycle, (c) Mixture of the acetone-in-water and methanol-in-water.....	40
Figure 3.9 Quality factor definition based on resonance frequency and 3-dB bandwidth.....	41
Figure 3.10 (a) 3D representation of recorded frequency, quality factor, and amplitude of various concentrations of methanol/acetone in water, which is projected in (b) amplitude vs. quality factor, (c) quality factor vs. frequency, and (d) amplitude vs. frequency	42
Figure 3.11 Temperature dependency of measured results is studied in simulation, considering the impact of substrate and MUT for water, methanol, and acetone.	43
Figure 4.1 Snapshot of the scattering parameter for single temperature	45
Figure 4.2 Scattering parameters stacked into single mega dataset including classes and temperature each recorded row belongs to	46
Figure 4.3 Representation of stacked frequency, amplitude, and quality factor (F,A,Q) with respect to the classes, labels, and temperatures	47
Figure 4.4 S_{21} for water and methanol at two extreme temperatures.....	49
Figure 4.5 Various concentration of methanol in water with irregular and nonlinear sensor output	50
Figure 4.6 Confusion matrix for material characterization with poor sensor stability showing worse prediction at lower concentrations of methanol in water due to lower loss leading to instability in sensor.....	52
Figure 4.7 Distribution of measured data of various methanol concentrations in water	54
Figure 5.1 KNN prediction scheme.....	59
Figure 5.2 (a) Error rate for various K values in KNN, (b) Confusion matrix for K = 7.....	60
Figure 5.3 Confusion Matrix for Decision Tree	61
Figure 5.4 Simplified MLP structure.....	62
Figure 5.5 (a) Confusion matrix for MLP, (b) Convergence curve for MLP	66
Figure 5.6 (a) Normal classification line for linear regression, (b) Deviation in predictive line of linear regression due to additional outlier	67
Figure 5.7 Sigmoid function with expressions	68
Figure 5.8 Fitting diagram for linear regression	71

Figure 5.9 Temperature cycle during experiments	72
Figure 5.10 Linear regression result for all classes	74
Figure 5.11 Machine learning flow from input data at varying temperature to reporting classified material with temperature.....	75
Figure 5.12 Dataset snapshot from Python	76
Figure 5.13 Predicted vs. Actual classes of 10 classes in use.....	78
Figure 5.14 Temperature prediction vs. actual values for 10 classes in use	78
Figure 5.15 (a) Confusion matrix for classification of methanol/acetone in water samples, (b) Temperature reporting for randomly selected test materials within all classes	79
Figure 6.1 Confusion Matrix for Decision Tree on lower concentrations	82
Figure 6.2 (a) Sweep on K number for optimum Mean Error, (b) Confusion matrix for K-Neighborhood in classifying 0-5 % methanol-in-water samples for K = 2	82
Figure 6.3 Schematic view of the hyperparametric study.....	84
Figure 6.4 Hyperparameter optimization study on the hidden layer.....	87
Figure 6.5 Hyperparameter optimization study with validation accuracy on neurons in the first and second layer.....	88
Figure 6.6 Hyperparameter optimization study on with validation loss on neurons in the first and second layer.....	88
Figure 6.7 (a) Hyperparameter optimization study on the batch size with (a) validation loss and (b) accuracy loss	89
Figure 6.8 Hyperparameter optimization study on the number of epochs with (a) validation loss and (b) accuracy loss.....	90
Figure 6.9 Hyperparameter optimization study on Kernel initializer with validation accuracy	91
Figure 6.10 Hyperparameter optimization study on Kernel initializer with validation loss	92
Figure 6.11 Hyperparameter optimization study on various optimizers, including Nadam, SGD, Adadelata, Adam, and RMSprop using validation accuracy.....	98
Figure 6.12 In-depth comparison of best optimizers, Adam vs. Adadelata, using validation accuracy in different number of epochs.....	99
Figure 6.13 Hyperparameter optimization study on various loss functions using validation accuracy	103
Figure 6.14 Schematic of the activation function	104
Figure 6.15 Hypermeter optimization study on activation functions, relu and elu, using validation accuracy.....	107
Figure 6.16 Hyperparameter optimization study on last activation functions, sigmoid and softmax, using validation accuracy	108
Figure 6.17 (a) Accuracy and loss of training and validation data, (b) Confusion matrix for optimized neural network resulting in 95 % accuracy.....	109

List of acronyms

AI	Artificial Intelligence
BP	Back Propagation
BW	Band Width
CPU	central processing unit.
CPW	coplanar waveguide
CSRR	Complementary Split Ring Resonator
DI	Distilled Water
DSVM	Decision-tree-based support vector machine
DT	Decision Tree
ELU	Exponential Linear Unit
ReLU	Rectified Linear
GNB	Gaussian naive Bayes
GPU	graphics processing unit
HFSS	high-frequency structure simulator
IDC	Interdigital Capacitor
IPA	Isopropyl Alcohol
KL	Kullback Leibler Divergence Loss
KNN	K-Nearest Neighbors
LC	Inductor capacitor
LOO	Leave-one-out
LR	Linear Regression
LUT	Liquid Under Test
MCSRR	Microstrip Complementary Split-Ring Resonator
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
MUT	Material Under Test
MW	Microwave

NAG	Nesterov-Accelerated Gradients
NN	Neural Network
PCA	Principle Component Analysis
RF	Radio Frequency
RH	Relative Humidity
RLC	Resistor Inductor Capacitor
RMS	Root Mean Square
SGD	Stochastic Gradient Descent
SKF	Stratified k-fold Class Validation
SRR	Split Ring Resonator
VNA	Vector Network Analyzer
WIFI	Wireless Fidelity

List of symbols

f	Resonance frequency
Q	Quality factor
A	Amplitude of S_{21}
ϵ_m	Dielectric constant of the material
M	Molecular weight
ρ	Density of the material
α	Molecular polarizability
N_A	Avogadro's number
μ	Dipole moment
k	Boltzmann's constant
g	Correlation factor between neighboring molecules
T	Temperature in $^{\circ}K$
$\epsilon_0(T)$	Static (low frequency) permittivity
$\epsilon_{\infty}(T)$	High frequency permittivity
$\tau(T)$	Relaxation time
$\tan \delta$	Loss tangent
$\nabla_{\theta} J(\theta)$	Gradient of the objective function
η	Learning rate
$J(\theta)$	objective function
b	Bias
$w_{to, from}^{layer}$	Weight
σ	Activation function

Chapter 1

Introduction

1.1 Motivation

Microwave planar sensors obtained profound importance in recent decades due to their capability in non-contact sensing, which enables many applications [1]–[8]. Among many features of these sensors, miniaturization is of great importance as the inspiration from metamaterial inclusion brings about reduced size at the frequency of interest [9]–[11]. Also, these sensors are found to be highly sensitive in detecting small quantities of contamination or solutes in air/aqueous solutions [3], [4], [12]–[19]. On top of all, radiation of microwaves into the air enables this type of sensors to reach out further distances compared with their competitive candidates including piezoelectric resonators (Surface/Bulk Acoustic Wave resonators [20]–[22]), electrochemical sensors [23], [24], waveguide-based microwave sensors [25]–[28], dielectric probe measurements [7], [29]–[33], etc.

It is noteworthy that the very efficacious feature of planar microwave sensors, non-contact sensing, can be viewed as their Achilles heel since this makes the sensor vulnerable to unwanted environmental variations. The way microwave sensors operate is to be affected either capacitively or resistively by environmental materials under test. This is done using a gap as the split of the ring resonators, which is found to be extremely receptive of small capacitance loadings from the environment [1], [34]–[42]. Having said that, perturbation theory used behind the theory of microwave sensors incorporates change in the effective electrical/magnetic energy in the environment and then converts that into a change

Chapter 1: Introduction

in either frequency of resonance or quality factor degradation [43]–[45]. It is then evident that the sensor has no selectivity over the sources of capacitance change. For instance, if the sensor is devised to keep track of environment relative humidity using a sensitive polymer layer on top of the resonator, it is expected to observe the changes in the adsorption/desorption of water molecules. But for all that, the sensor is exposed to variable temperature in the environment as well. This is an inevitable condition for most industrial sensory applications. It can be shown that the temperature also impacts the effective electrical properties of the substrate that holds the microwave resonator [46]. One immediate implication of unknown temperature variation is a drift in the sensor response only due to the temperature. Depending on the significance of these variations, the observation of relative humidity might be deviated or even overridden, which in many cases can be extremely troublesome and expensive to compensate.

There are many other examples where the sensor is under ambient sources that are not exact parameter under study [47], [48]. In biomedical studies, for example, the glucose level in human blood is crucially important for diabetes since any disorder in its level, whether higher or lower than normal, could lead to significant health issues, and even death in some cases. Conventional method of pricking the finger in the hope of a blood sample is not convenient for the children and the elderly. Yet, this process is often sought to be continuous to improve the patients' well-being. Microwave sensing has shown promising results to replace the current time-consuming and expensive check-up procedure since GHz frequencies can travel through human skin and flesh and reach the vessels at some parts of body such as fingertips. However, glucose content in human blood combined with many other parameters including proteins, ions, gases, wastes, etc. inside the plasma and red blood cells can affect the sensor response. This shows a potential challenge in attributing the sensor response to only desired parameter. Therefore, we are after new methodologies to marry microwave sensing platforms to enhance their capabilities.

1.2 Objectives

As of the first decades of 2020, industries and many research teams have endorsed the role of artificial intelligence (AI) in human life. AI simulates the real conditions with respect to a learning curve, which is highly dependent on the number of examples one can feed the learning system with. This technology has potential to discern linear and nonlinear patterns in the natural behavior of human beings or even man-made appliances. As a result, one can train a network to manipulate the operation of a real system. The benefit of this entire scheme is to improve the functionality of present systems with the intelligence and confidence of an unsupervised machine that yields predictions of system responses with the goal of optimizing the output. In recent years, the weather forecast, for instance, has been privileged from years of input data into predicting new patterns in the upcoming daily weather [49], [50]. It has the potential to provide suggestions to customers what to choose when they are shopping online. AI can also customize presentation of social media information to individuals based on their interests, according to their previous likelihood patterns. Moreover, today's world seeks harmony between various devices and platforms within a broad spectrum of technologies called Internet of Things. AI has tremendous potential to support this process with smart solutions including connectivity, energy harvesting [51]–[56], etc.

This technology with such vast range of potential application is also targeted in this thesis to aid microwave sensors. The way it is expected to work is to learn the patterns of sensor responses in correlation with many parameters. The complex interconnections of neural network, for one, can interpret large datasets and resolve the relationship between the final outcome and features of sensor response. In the proposed thesis, this capability is examined using several algorithms and a comprehensive study of the various techniques is performed with the goal of optimum outcome.

1.3 Thesis Outline

This thesis is outlined as follows:

Chapter 1 starts with WHY this study has been initiated, with posing the questions that have been raised in microwave sensor technology with respect to complex environmental sensing platforms. This chapter also suggests the objective of the effort put into this thesis.

Literature review of possible techniques to remove external unwanted noisy sources is given in chapter 2. It shows that with the recent achievements according to the reported literature, there is still room for improvement in this area. Also, the recent developments in AI that might be helpful for our needs are also discussed.

Chapter 3 describes the type of microwave sensor used in this work and the applications that this type of planar sensor can fit into. This chapter starts with the basic discussions on the sensor response and meaningful interpretation of its features. Then, the problem with this sensor is elaborated when exposed to uncontrolled environmental variations, in this case temperature, with analytical expressions. This chapter ends with experimental preparations including the materials used in this study.

Datasets from experiment need to be prepared in order to be fed into the network. Proper steps for combining the test data and avoiding possible anomalies in data gathering are explained in chapter 4.

Chapter 5 includes several machine learning algorithms that are used in this study including Decision Tree, K-neighborhood, and Multilayer Perceptron. Analyzing all input dataset with various learning algorithms lead us to neural network as the most versatile and effective method of resolving issues of environmental interference. Moreover, this study links a linear regression with neural network in a unique scheme to predict the temperature of environment, beside removing its impact on the sensor response for possible further post processing of data.

In-depth analysis and comprehensive study of neural network with deciphering all its internal functions and parameters are presented in Chapter 6. The use of neural network led to optimal design of temperature compensation with accuracy

Chapter 1: Introduction

of 95 % in confusion matrix for classifying the materials under test and 100 % temperature reporting accuracy.

This thesis concludes with chapter 7 with information regarding possible challenges in present microwave sensors that are resolvable using AI in future research work.

Chapter 2

Literature Review and Background

In modern sensing, access to inaccessible materials in industrial environment, flexibility in material type to monitor, and longevity of the sensor under use have been the moot points to discuss for a long time. It, then, became extremely attractive era for microwave community to find split-ring resonator designs quite easy to use when it comes into measuring wide varieties of materials at various environmental conditions. These sensors are privileged with the non-contact features of sensing that allow them to be used ubiquitously [57]–[61]. Treating microwave sensors has yet become predominantly challenging due to their vulnerability to undesired effective impact from the environment, beside desired effects. Among them, relative humidity is an easy example to follow [62], [63]. Imagine a planar sensor with full access to the surrounding environment that is set up to measure the temporal behavior of a given solution inside a tubing/microfluidic channel mounted directly on the sensor. This, beside providing us with benefit of remote sensing, has drawbacks of being sensitive to the level of relative humidity (RH) in the air surrounding the sensor since the air with RH has effectively higher dielectric constant compared with dry air. Therefore, the impact on the sensor's output is accumulation of material and ambient RH, which results in erroneous outcome.

The effective factors on the sensor's result depends on the detection methodology and the principle of operation, which in the case of microwave planar sensors this includes relative humidity, temperature, pressure, material proximity, and displacement. Among these, temperature impact on the planar sensors is the main focus of this thesis, where detrimental implications of extra temperature are shown to develop anomalies in the characterization stage. This effect can be as large as confusing a decision-making system in discerning the type of material. Also, this

may be quite troublesome in distinguishing mixing ratios when it comes into known mixtures.

In this section, it is instructive to review some methodologies in microwave domain that are potent in resolving this issue. Through this, our proposed machine learning algorithm is explained with respect to all additional and comprehensive benefits compared with the bulky and complex designs. With this in mind, the following sections elaborate reference-sensor based technique, which intuitively controls the sensor's behavior using an additional reference sensor.

2.1 Dumbbell-Shaped Defect Ground Structure

Another type of calibration method as a referencing method for microwave planar sensor design is given in [64] using dumbbell-shaped defect ground structures. This sensor, apparently different in the preface, is similar to the previous MCSRR example in the way it compensates for external effects using a reference sensor besides a measurement sensor. This sensor, however, uses a dumbbell-shaped resonator, again etched out of the ground plane that has sensitive regions at the two end patches, as shown in Figure 2.1. Moreover, the inference technique in this setup is computing the cross-transmission feature of the combined system that is ideally null in case of full symmetry between the two sides, including the channels and materials inside. In contrast, with a change in the material type, the symmetry breaks and the sensor output demonstrates a non-zero result of the following computation:

$$|S_{21_{cross-mode}}| = \frac{1}{2} |S_{21} - S_{43}| \quad 2.1$$

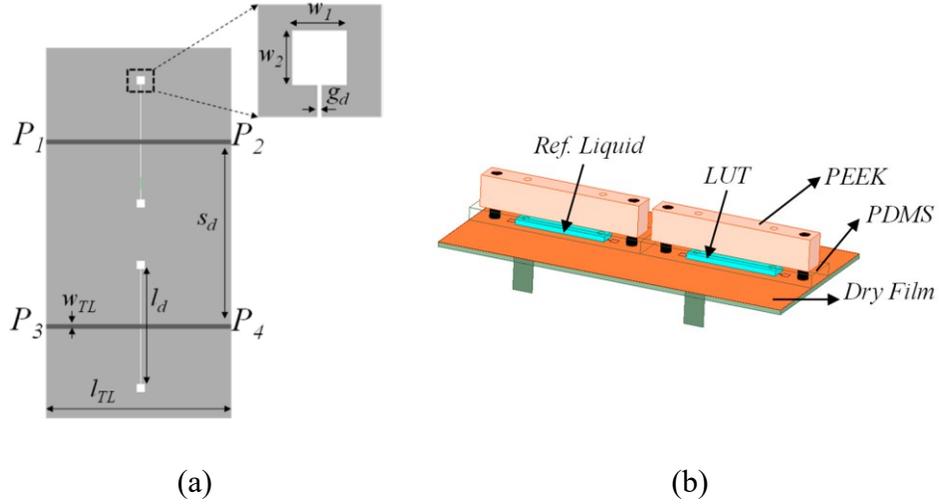


Figure 2.1 (a) Dumbbell-shaped resonator design with two sensing regions for measurement and reference sensors, (b) Microfluidic channels mounted on the planar resonator, all excerpt from [64]

2.2 Transmission Lines Terminated with *LC* Resonators

This resonator is based on the series RLC [65] that is materialized with a patch (as capacitance), via (inductance), and chip resistor (as a resistor), as shown in Figure 2.2(a). The resonator is repeated identically to result in two adjacent resonators. One is used to measure the dielectric constant of solid pads with fixed thicknesses that are directly placed on the resonator's patch as the most sensitive capacitive region. The reference sensor is also covered with TMM4 ($\epsilon_{r-1} = 4.7$) a default dielectric slab while the other resonator is to sense dielectric constant values within the range of $\epsilon_{r-2} = 1.88 - 11.28$. The sensor outputs two resonance frequencies corresponding to each resonator as shown in Figure 2.2(c) where the f_{r1} represents the reference frequency and f_{r2} is monitoring the materials on the sensing resonator. Also shown in Figure 2.2(c) is the sensitivity of the sensor, defined as follows:

$$S = \frac{\Delta f_{r2}}{\Delta \epsilon_{r2}} \quad 2.2$$

and represented with respect to differential permittivity values of $\epsilon_{rd} = \epsilon_{r2} - \epsilon_{r1}$.

2.2

This method considers the level of variation from a reference slab. It reports the variation as the sensor value, which includes all environmental changes as well. Yet, these variations might be different for various slab dielectric constants. Therefore, this scheme would be quite confusing if the exact environmental variations are not studied in advance on the default resonator and its default slab. Although this method is presented for solids, it has the potential to be employed for liquids as well. In both cases, the sensor, as well as the materials, are all impacted by temperature variations, which can obscure the absolute impact on the material under test. In other words, there might be a case where the reference material impacted by the temperature causes reference frequency to change slower than the material under test due to different temperature expansion coefficient. As a result, this method also has complications regarding accurate temperature-based recognition.

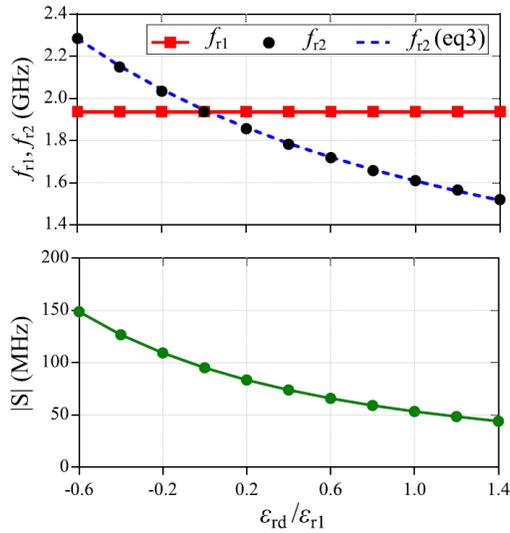
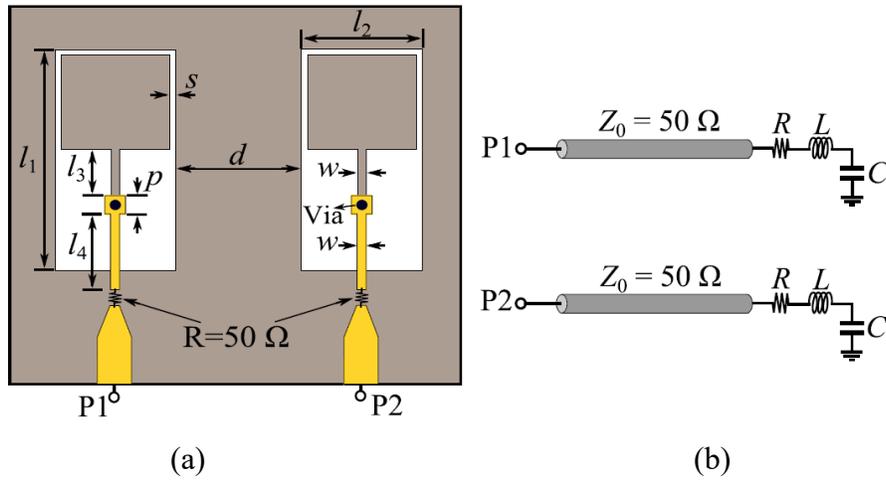


Figure 2.2 (a) Schematic of transmission line terminated with LC resonator, (b) electrical circuit equivalence of the resonator, (c) Frequency changes in the sensing resonator with sensitivities; all excerpt from [65]

2.3 Rat-race based differential Sensor

This method of sensing is more recent in microwave sensing algorithms that are based on the constructive and destructive summation of power in one port as in [66]. The incoming power into the output port is differential between the reflections from two resonators: one reference and one measurement resonator. In simpler words, the input power of port-2 (Δ) to the rat-race is designed to reflect from port 3 (ρ_3) and 4 (ρ_4) and combined in port-1 (Σ) as articulated below (see Figure 2.3(a)):

$$S_{12} = -\frac{j}{2}(\rho_3 - \rho_4) \quad 2.3$$

The two resonators are separated from the main rat-race circle with transmission lines in order to avoid unnecessary interference between them. Similar to previous structures, two sensing and reference channels are mounted on top of the sensing regions (resonators). The fabricated sensor is shown in Figure 2.3(b).

The sensor is injected with IPA mixed with DI water with various ratios between 0 – 100 % with 10% increments. The sensor output demonstrates the lowest transmission from port 2 to port 1 (S_{12}) when the two reflections from ports 3 and 4 are identical, which means the material under test is the same as the reference, which is DI water in this case. The higher the IPA ratio, the more different ρ_3 and ρ_4 become.

The sensing mechanism described in this paper, although different from the previous resonator-based sensors, is still impacted by the temperature variations of the reference liquid as well as other potential problems mentioned with the previous sensors. Thus, if the temperature expansion coefficient of the material under test and the reference material are not quite similar, the sensor would be detuned, and the outcome cannot be compensated for temperature.

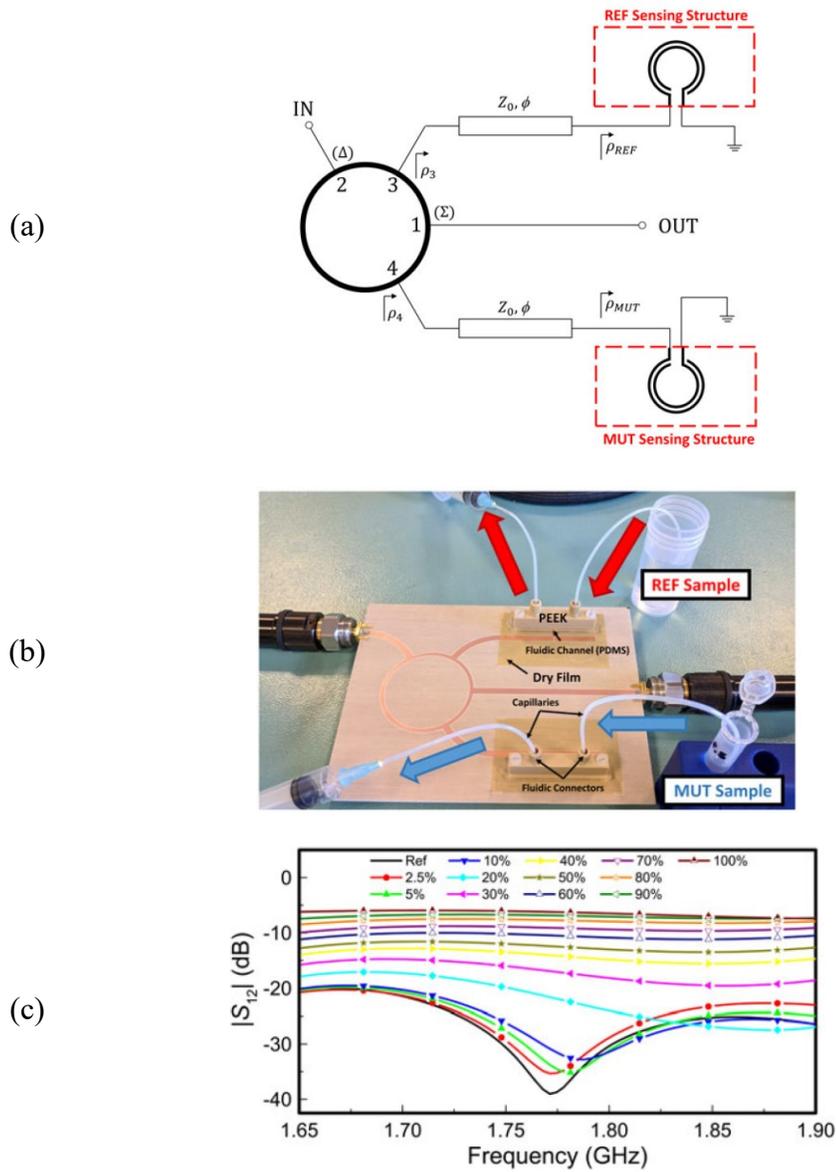


Figure 2.3 (a) Schematic of rat-race based reflective-mode differential sensor, (b) Measurement scenario, (c) Output of the sensor in discriminating various concentrations of IPA in DI water; all excerpt from [66].

2.4 SRR-based relative humidity compensation scheme

This method is introduced to make a real and impactful potential application using microwave SRR-based resonators in removing the impact of relative humidity (RH) of sand. High levels of RH in the medium might severely attenuate the transmission signal such that the resonance profile might die out completely. The authors in [48] used an efficient technique to recover possible lost resolution in transmission profile using positive feedback. In this method, SRR acts as the frequency selective unit of a feedback system with a high gain amplifier that is set with proper phase aggregation through the loop to reconstruct the lost power. This allows for the sensor to operate in highly lossy medium without being impacted by uncontrollable variations of the RH in the environment.

In this specific example, two identical microwave planar sensors at ~ 1 GHz and ~ 1.15 GHz are designed in parallel. One resonator is mounted with a tubing to cover the sensitive region, which holds the material under test. The other resonator is left in direct contact with the environment. Next, the whole sensor system is covered with 10 cm of sand, where the tube holding MUT is at 1 cm above the sensor surface, as shown in Figure 2.4(a). It is shown in Figure 2.4(b) that the sensing (f_1) and reference (f_2) frequencies are both impacted by RH, although not quite the same. The relationship between downshifts of f_1 and f_2 is extracted for known RH levels of 20%, 40%, 60%, and 80% with material inside the tube (permittivity values of 10, 20, and 30) using HFSS simulations as shown in Figure 2.4(c). Then, the whole system is analyzed with samples of water/ethanol/methanol inside the tube and the sand is wetted using uniformly dispensed water droplets as a means to increase its RH level. It is shown that, with a proper compensation algorithm, one can restore the sensor's response to given materials without dependence on ambient RH.

This method, even though really applicable to RH, seems to face drawbacks when it comes to measuring material change with temperature impact since materials themselves change their dielectric constants and this change is not compensated in any way.

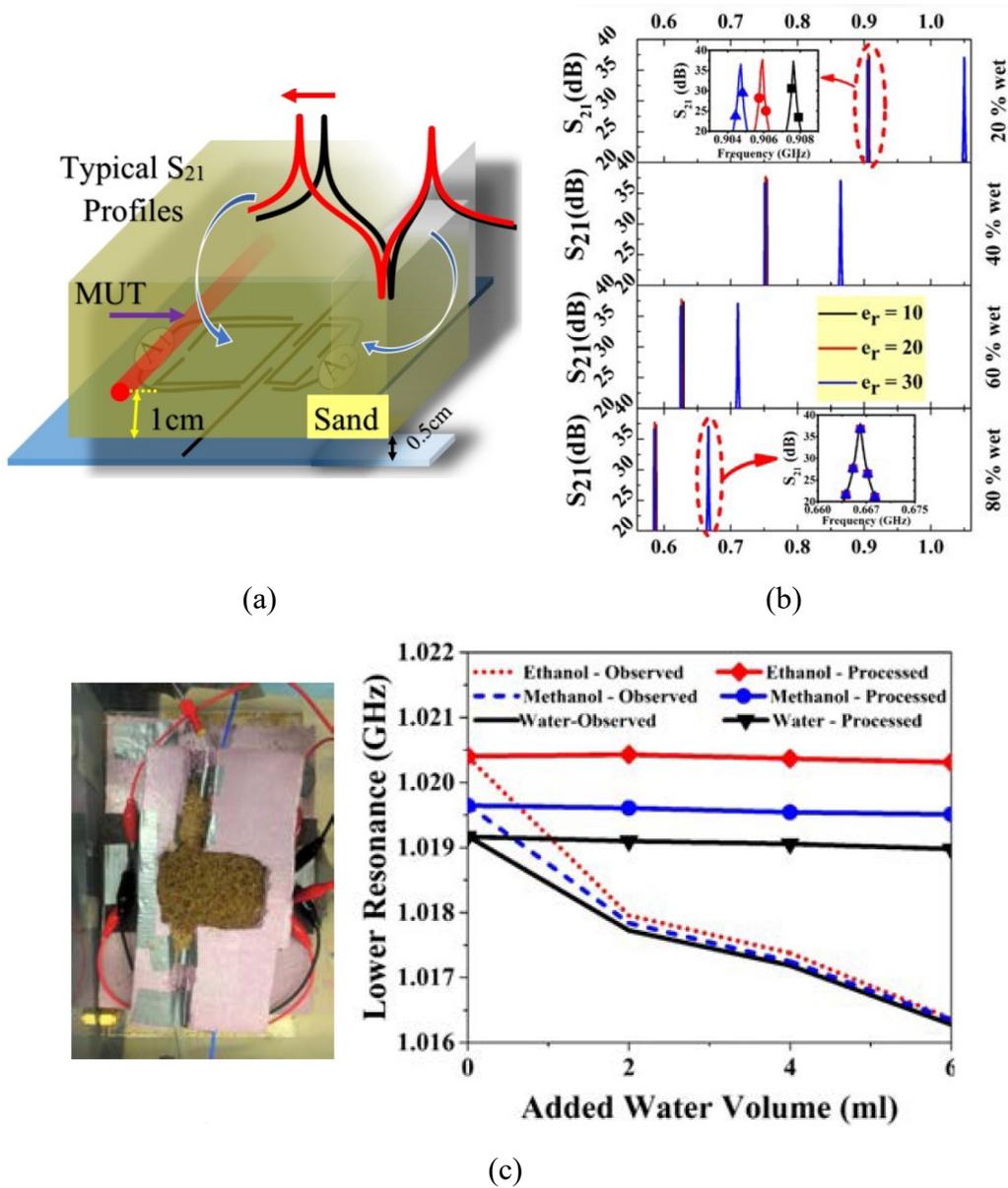


Figure 2.4 (a) Schematic of double loss-compensated SRR sensor covered with sand, (b) Both reference and sensing resonator are impacted by the RH, (c) Sensor is covered with sand experimentally and methanol/ethanol/water are calibrated to recover regardless of the applied RH; all excerpt from [48].

2.5 Dual-mode SRR to eliminate RH

An innovative method of sensing without being influenced by environmental RH impact is introduced in [47] with a design change in the topology of the SRR as shown in Figure 2.5. The idea lies in using the first ($f_1 = 552 \text{ MHz}$) and second ($f_2 = 2f_1 \sim 1.1 \text{ GHz}$) harmonic of the SRR. The HFSS simulation results at first harmonic (left) and second harmonic (right) are shown in Figure 2.5(a). The location labeled C is an important feature, because it is highly insensitive with respect to the 2nd harmonic (since it lies exactly at $\lambda/8$ from each end of SRR making this location as a null for the second harmonic) and sensitive to the 1st harmonic (since it is away from the middle of SRR that acts as a null for 1st harmonic). This location is engineered to hold both features of affecting 1st harmonic and not affecting 2nd harmonic simultaneously. Therefore, it seems an ideal location to monitor only MUT using 1st harmonic while being impacted by RH on 1st harmonic and compensating for the excessive RH impact by properly monitoring the modified 2nd harmonic that is unaffected by MUT but changes only due to RH. In this example, it is shown in Figure 2.5(b) how various sensor's response with materials inside the tube is impacted by RH level varying between 5% – 70%. The resonator shows no considerable change in resonance frequency due to the material inside tube without RH in Figure 2.5(c). However, these variations become strongly modified when the sensor is exposed to high RH levels, where the absolute shifts of resonance frequency for sensor holding specific MUT is given in Figure 2.5(b).

Although this method has an innovative and compact design for compensating RH at low frequencies, it is not invincible when it comes to compensation for temperature. This is because the MUT's dielectric properties also change with temperature and the produced error is not resolved.

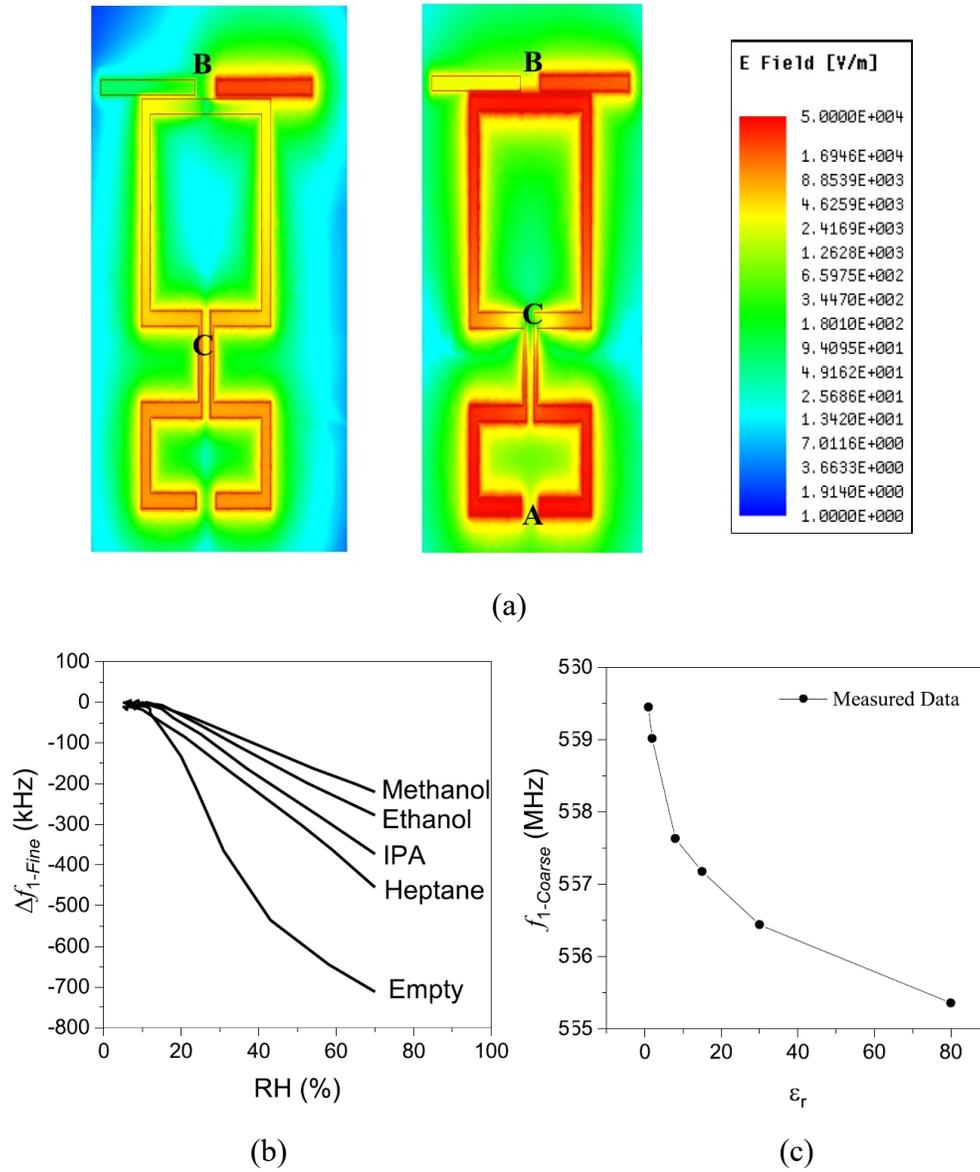


Figure 2.5 (a) Dual-mode microwave planar sensor, (b) variation in sensor response when various MUT are injected into sensor with varying RH, (c) Sensor response to permittivity range of 1-80 without RH; all excerpt from [47].

2.6 Dual Mode Microwave Microfluidic Sensor

This study has attempted to employ a reference resonator to monitor the temperature of the environment in [67]. The resonator is an open-ended half-wavelength resonator that is enclosed by metallic packaging. The resonator is mounted with a capillary tube on one resonator to allow liquid passage on the sensing resonator. Both resonators are designed at close-by frequencies ~ 2.5 GHz (see Figure 2.6(a)). The principle of operation of this sensor is to record the frequency/quality factor variation in the reference sensor with empty capillary as follows:

$$\begin{aligned}\Delta f_{ref}(T) &= f(T) - f(25^\circ) \\ \Delta Q_{ref}(T) &= Q(T) - Q(25^\circ)\end{aligned}\tag{2.4}$$

The corrected values for the measured frequency and quality factor will then be the subtraction of the measured frequency from resonator holding the material under test and the corresponding variation in the reference resonator ($\Delta f_{ref}/\Delta Q_{ref}$) as below:

$$\begin{aligned}f_{corrected} &= f_{measured} - \Delta f(T) \\ Q_{corrected} &= Q_{measured} - \Delta Q(T)\end{aligned}\tag{2.5}$$

This results in correction of the extraneous impact caused by the sensor and capillary only since these are embedded in $\Delta f_{ref}/\Delta Q_{ref}$. As shown in Figure 2.6(b-c), the change in resonance frequency and quality factor is not removed and not even decreased. However, is recovered and corrected to a case where the sensor reports the exact variation in the response only related to the material under test.

It is interesting to note that this article has claimed a method of removing only the sensor's response, yet the problem with the erroneous impact of temperature on incorrect recognitions is still present. In other words, imagine a case when the material inside the tube is unknown, or it can be selected among the few known

Chapter 2: Literature Review and Background

cases, even when the main material is mixed with another solute. Then, this method fails to properly recognize the type of material since the resonance frequency and also the quality factors of different materials may overlap when they are observed at various temperatures. This leads to inaccuracy in either material selection and also the temperature of the environment since the referencing method is not potent in removing the change in dielectric constant of material due to temperature. Therefore, the corrected results in resonator due to referencing still hold the unwanted sensor response due to temperature.

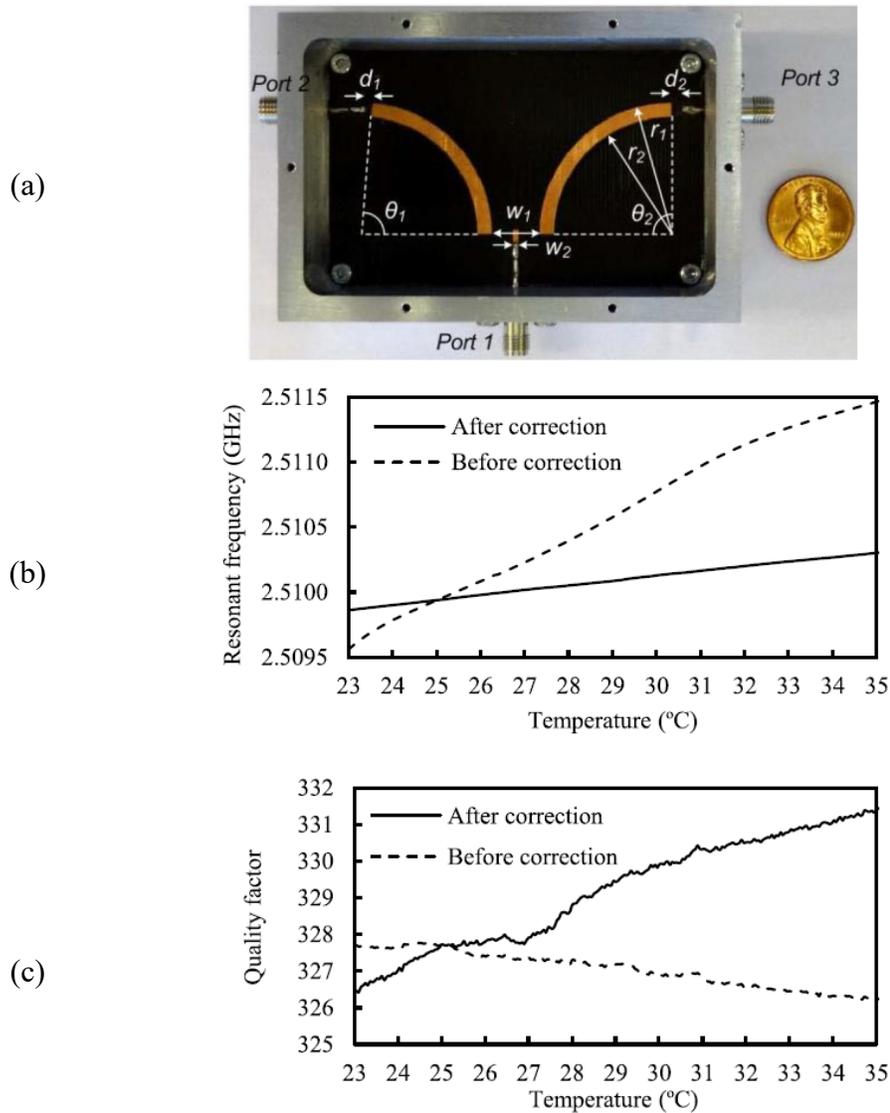


Figure 2.6 Dual-mode resonator design, (b) Chloroform corrected resonant frequency and (c) quality factor with respect to temperature; all excerpt from [67]

This theme of studying the microwave sensors is also shown in a similar paper [68], wherein water is used as the sample. Again, a reference resonator is used to monitor the temporal behavior of sensor response for temperature variation within 20-40°C. However, the corrected dielectric constants still convey a trend inferred from the sensor that is varying with respect to the temperature, as shown in Figure 2.7.

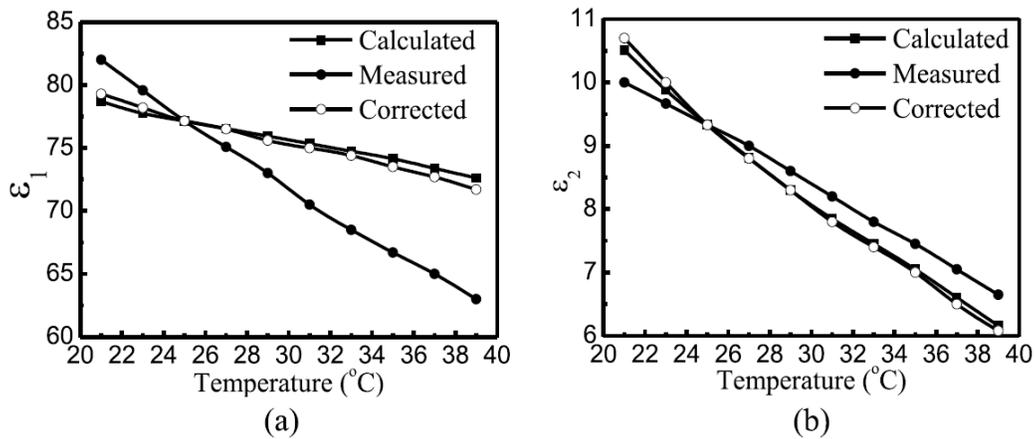


Figure 2.7 (a) Real and (b) imaginary part of the permittivity of water when measured and corrected with respect to temperature change; all excerpt from [68]

This is the main problem with microwave sensors that hold the impact of temperature since it impacts directly on the dielectric constant. It will be deeply discussed in the next chapter regarding the potential problem of planar sensors to discriminate between different materials at varying temperatures, while the presented methodologies are not successful in resolving the problem.

2.7 Symmetric CPW Sensor with IDC for permittivity characterization

This article represents a method to extract the complex permittivity of unknown material using an interdigital capacitor (IDC) based symmetric coplanar waveguide (CPW) sensor [69]. As shown in Figure 2.8(a-b), the sensor is composed of two IDC units that are engraved from the central conductor (signal line) of the CPW. The sensor is designed to work in the operational resonance frequency at 4.5 GHz. Since the system is symmetric, two 3 dB power dividers are used to transfer the power to both IDCs. In this work, both resonance frequency shift Δf and amplitude of S_{21} are fed as input data to the neural network (shown in Figure 2.8(c)) trained using backpropagation (BP) algorithm. The structure of the network is composed of two neurons as input layer (one for Δf_r and one for $\Delta|S_{21}|$), followed by one hidden layer towards an output layer (with one neuron).

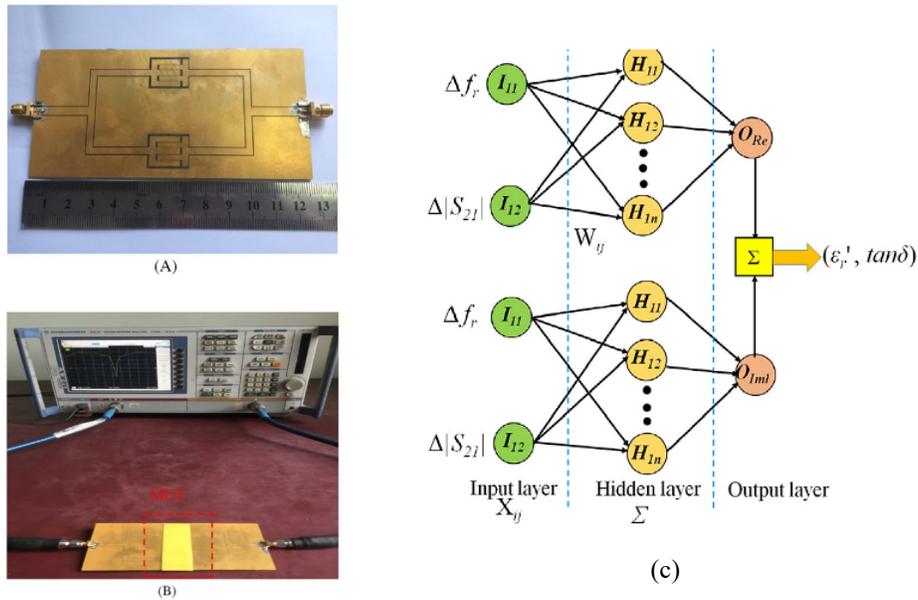


Figure 2.8 (a) Photo of the fabricated sensor, (b) Experimental setup, (c) Artificial neural network structure for permittivity extraction; all excerpt from [69]

The BP algorithm runs two times, first, to provide the real part of the dielectric constant, and next to extract its imaginary part. Both resonance frequency and amplitude of S_{21} are highly dependant on the real and imaginary part of the dielectric constant as shown in Figure 2.9. Thus, it is a wise decision to consider these two features as the most critical to feed the network. However, the data are collected in a completely controlled and ideal environment, i.e. there is no type of error source. Each possible error (such as temperature and relative humidity changes, material displacement on top of the sensor, etc.) results in considerable changes in both Δf_r , and $\Delta|S_{21}|$. Therefore, there is a massive deviation on their values such that the overlap of data makes the training very poor.

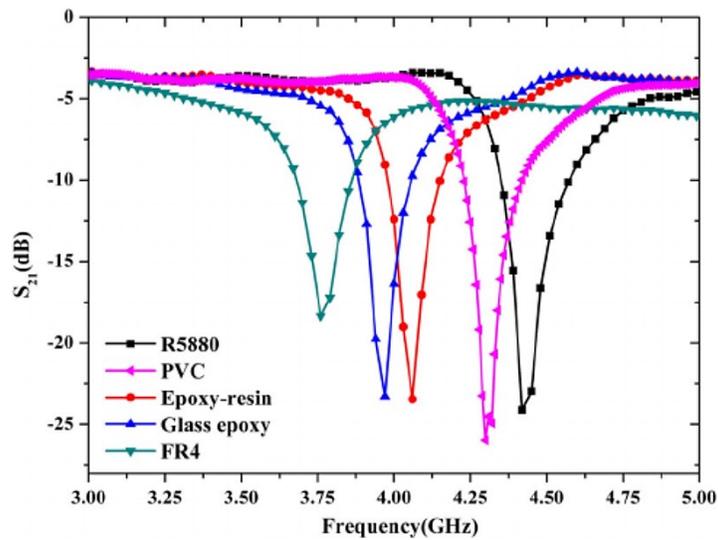


Figure 2.9 The measured frequency shift and amplitude value change for various materials; excerpt from [69].

2.8 Material Identification Using CSRR Array and Machine Learning

In this study, an array of split ring resonators, each with a specific resonance frequency, is used to classify three types of materials, including cardboard, wood, and plastic samples with the aid of machine learning algorithms, as shown in Figure 2.10(a) [70]. Changes in dielectric constant are covered over a wide range of frequencies from 1 GHz to 10 GHz due to the combination of different resonators as array. The resonance frequency notches occur at 1 GHz, 3 GHz, 5 GHz, 7 GHz, and 9 GHz. Thus, change in dielectric constant results in the downshift of these resonance frequencies, as shown in Figure 2.10(b).

These five resonance frequencies and their combinations, meaning resonance sift from (sensors 1 and 2) or (sensors 2, 3, and 4) result in 31 combinations, are considered as input data to the network.

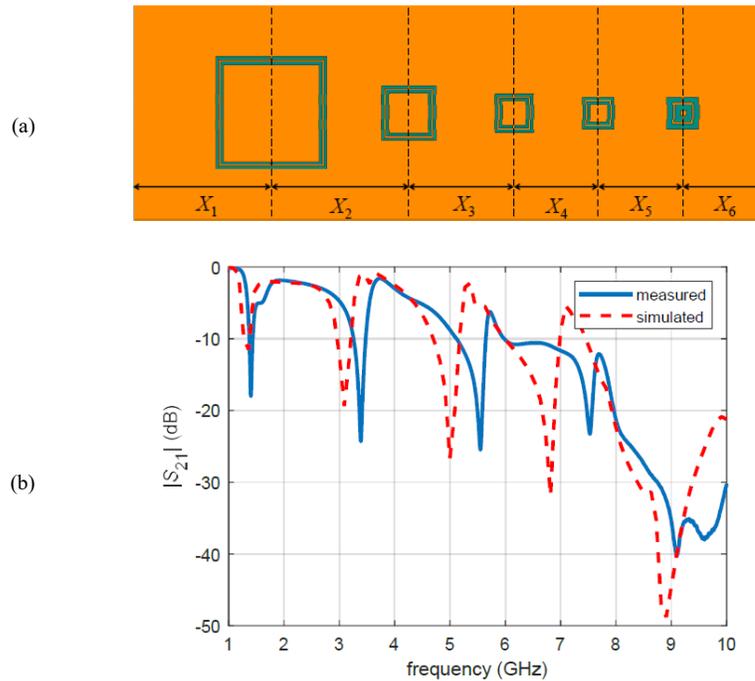


Figure 2.10 (a) Array of CSRRs with frequencies of 1.36 GHz, 3.09 GHz, 5 GHz, 6.82 GHz, and 8.91 GHz, (b) Resonance frequencies of the array; all excerpt from [70]

Six different algorithms are studied to classify the materials as follows: Decision tree (DT), Decision-tree-based support vector machine (DSVM), k-nearest neighbors (KNN), Random Forest (RF), Gaussian naive Bayes (GNB), and Multilayer Perceptron (MLP). Here, the two procedures of Leave-one-out (LOO) and stratified k-fold class validation (SKF) are used to evaluate the performance of the algorithms. After classification with a different algorithm, results for cardboard classification show that DSVM has better performance with both LOO (Figure 2.11(a)) and SKF (Figure 2.11(b)) cross-validation (86.4% of accuracy at best) when using the resonance frequency shift of sensors 1, 2, and 5 as selected features. This paper indeed introduces a similar method to [71], which is broadband spectroscopy of materials with a more sensitive approach using split-ring resonators. Besides its capability in classifying various solid slabs, all of the presented resonators are prone to deviating impact of unwanted environmental sources.

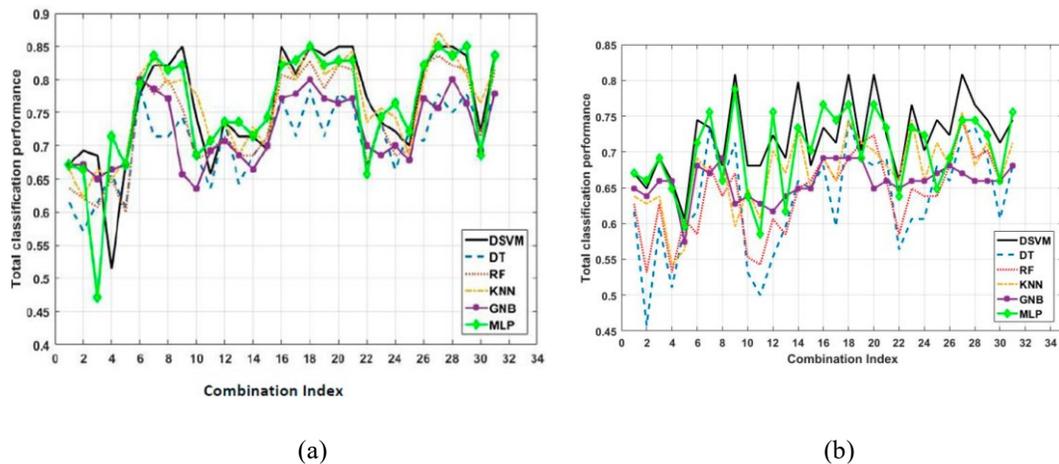


Figure 2.11 Total classification accuracy versus combination index for classifying cardboard using (a) LOO- and (b) SKF-cross-validation, all excerpt from [70]

2.9 Microstrip Complementary Split-Ring Resonator (MCSRR)

Complementary structures are etched designs from the ground layer of the planar sensor that holds the resonant structure's complimentary, as shown in Figure 2.12 as given in [72]. The interrogating microstrip line lies at the other side of the substrate to only couple input power and elicit the output power through the coupling.

In this sensor, one of the two resonators are used for measuring materials inside the microfluidic channel on the resonator, while the other resonator is used as a reference. The operation principle for the sensor is to measure the reflection coefficient (S_{11}) from each resonator, comparing which would result in an effective presence of the material inside sensing microfluidic channel. The two ports are connected to the resonators from one side only, while the second ports are terminated with 50-Ohm resistances in the middle of the design (see Figure 2.12(a)). The sensing resonator is tested with dielectric constant values ranging from 1-80 at two different loss-tangent parameter values of $\tan \delta = 0, 0.2$. The sensing resonator depicts a frequency variation from 1.6 GHz down to 1 GHz and amplitude change from ~ 22 dB up to ~ 10 dB (see Figure 2.12(b)). On the contrary, the reference sensor stays still even with holding similar microfluidic channel on top since the two sensors are completely uncoupled (see Figure 2.12(c)).

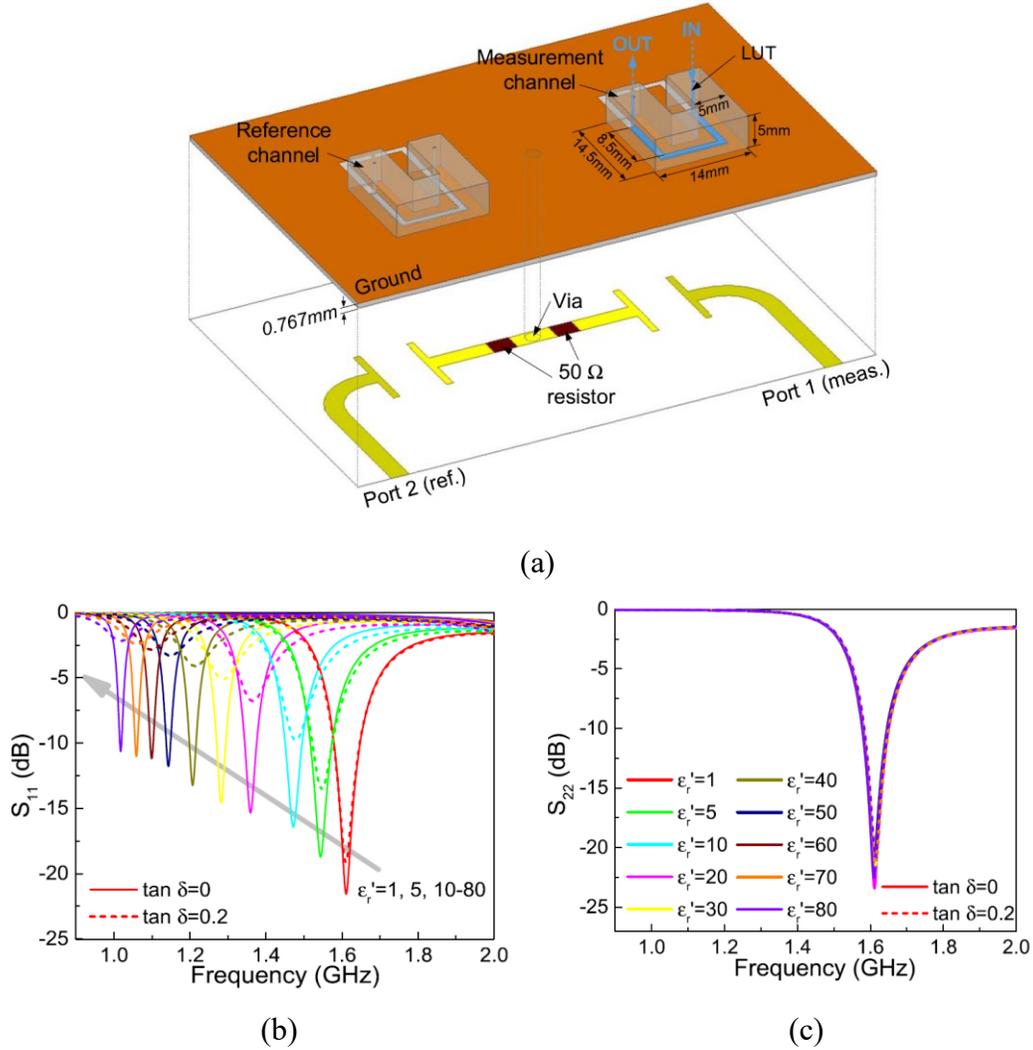


Figure 2.12 Microstrip Complementary Split-Ring Resonator loaded with microfluidic channels, (b) Variation in the first resonator when the material inside varies, (c) Stability of reference resonator regardless of the material variation on sensing resonator; all from [72]

This sensor has the following approach towards dielectric constant assessment. First, in order to extract the real permittivity (ϵ_r'), the relative frequency shift is used for characterizing the liquid under test (LUT) as follows:

$$f_r = \frac{f_{ref} - f_0}{f_{ref}} \quad 2.6$$

$$\epsilon_r' = \alpha e^{f_r} - C \quad 2.7$$

where f_0, f_{ref} , and f_r are resonance frequency of sensing resonator, reference resonator, and relative change in resonance frequencies. Next, a backpropagation neural network (BP-NN) with three layers (one hidden layer) is used to provide the loss tangent that accepts real permittivity value of ϵ'_r , and the normalized quality factor $Q_{nor} = \frac{Q_{ref}}{Q_0}$ as input, see Figure 2.13.

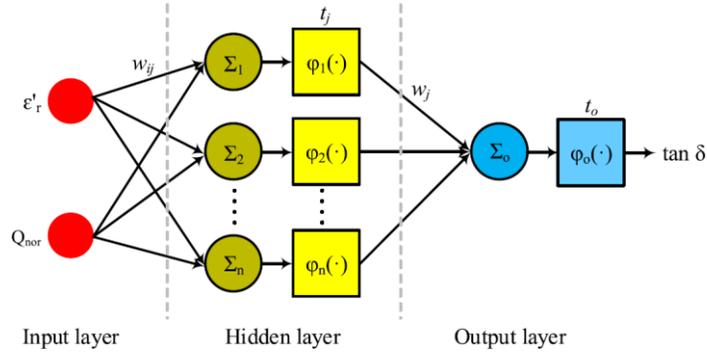


Figure 2.13 Structure diagram of the BP-NN, [Excerpt from [72]]

Differential sensors are commonly used to remove the environmental effect on sensors. Environmental errors such as relative humidity, displacement of materials under test, pressure, etc., do not change the properties of materials, but only the sensory setup. It is worth to note that temperature as an error, however, is exceptionally different from other types of error sources since it affects the material properties directly and changes the dielectric constant of the material. Here to clarify this claim, let us consider the following schematic of the proposed sensor. The resonance frequency of reference resonator, as shown in Figure 2.12 and Figure 2.14 with microfluidic channel is f_{ref} , and the sensing resonator with microfluidic channel and material is $f_0 = f_{ref} + f_m$, where f_m denotes only the extraneous effect of the material. If the temperature changes during the experiment, each resonance frequency is shifted to f'_{ref} , and $f'_{ref} + f'_m$. It can be inferred from this paper [72] that the temperature impact could be removed by subtracting the updated sensing and reference resonance frequencies at a given new temperature. However, it will

be shown during the thesis that f'_m is not equal to the f_m while the effect of temperature, originated from the sensor by itself, is omitted.

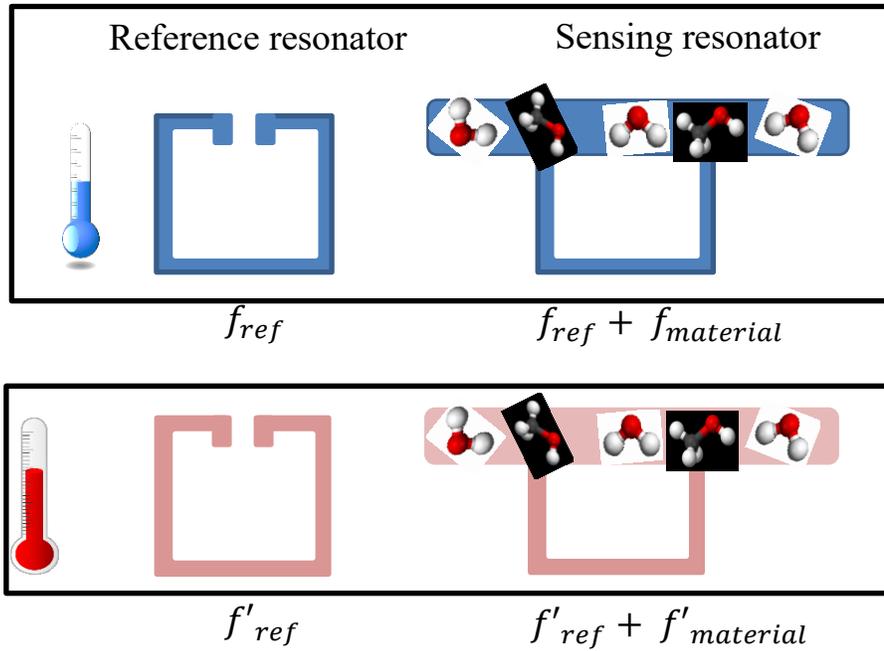


Figure 2.14 Schematic of the sensor exposed to MUT in varying temperature

$$f_{ref} + f_{material} - f_{ref} = f_{material} \quad 2.8$$

$$f'_{ref} + f'_{material} - f'_{ref} = f'_{material} \quad 2.9$$

$$f_{material} \neq f'_{material} \quad 2.10$$

In other words, even though the reference sensor changes due to temperature as well as the measurement sensor, this doesn't help to create a new reference since the material inside the measurement microfluidic channel changes into something new. That's why the problem with incorrect recognition of the material type still holds true with this level of sensor design.

Chapter 3

Microwave Sensor Design

3.1 Scattering Parameters of the Sensor

The proposed microwave sensor is a split-ring resonator, which is the core of the sensing platform. The dimensions of the sensor are given in Figure 3.1, where it is noteworthy that the design of the coupler between input/output transmission lines and the SRR is narrow to increase the inductance of the resonator and hence reduce the frequency of operation that means the design is miniaturized.

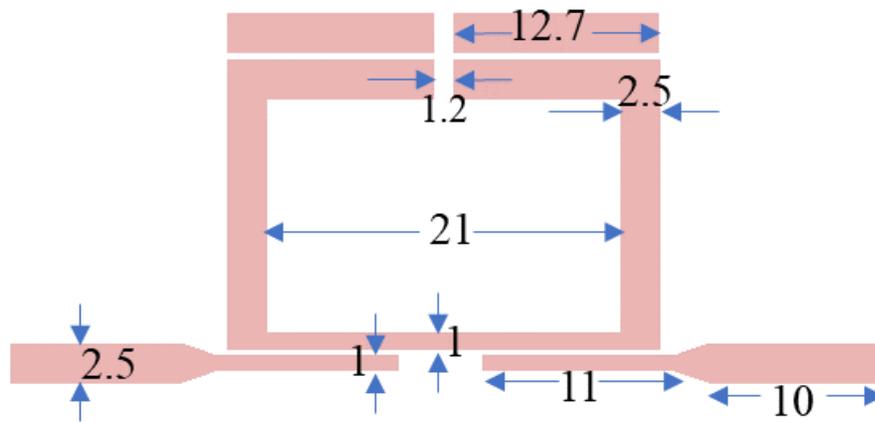


Figure 3.1 Schematic of SRR with all dimensions in [mm].

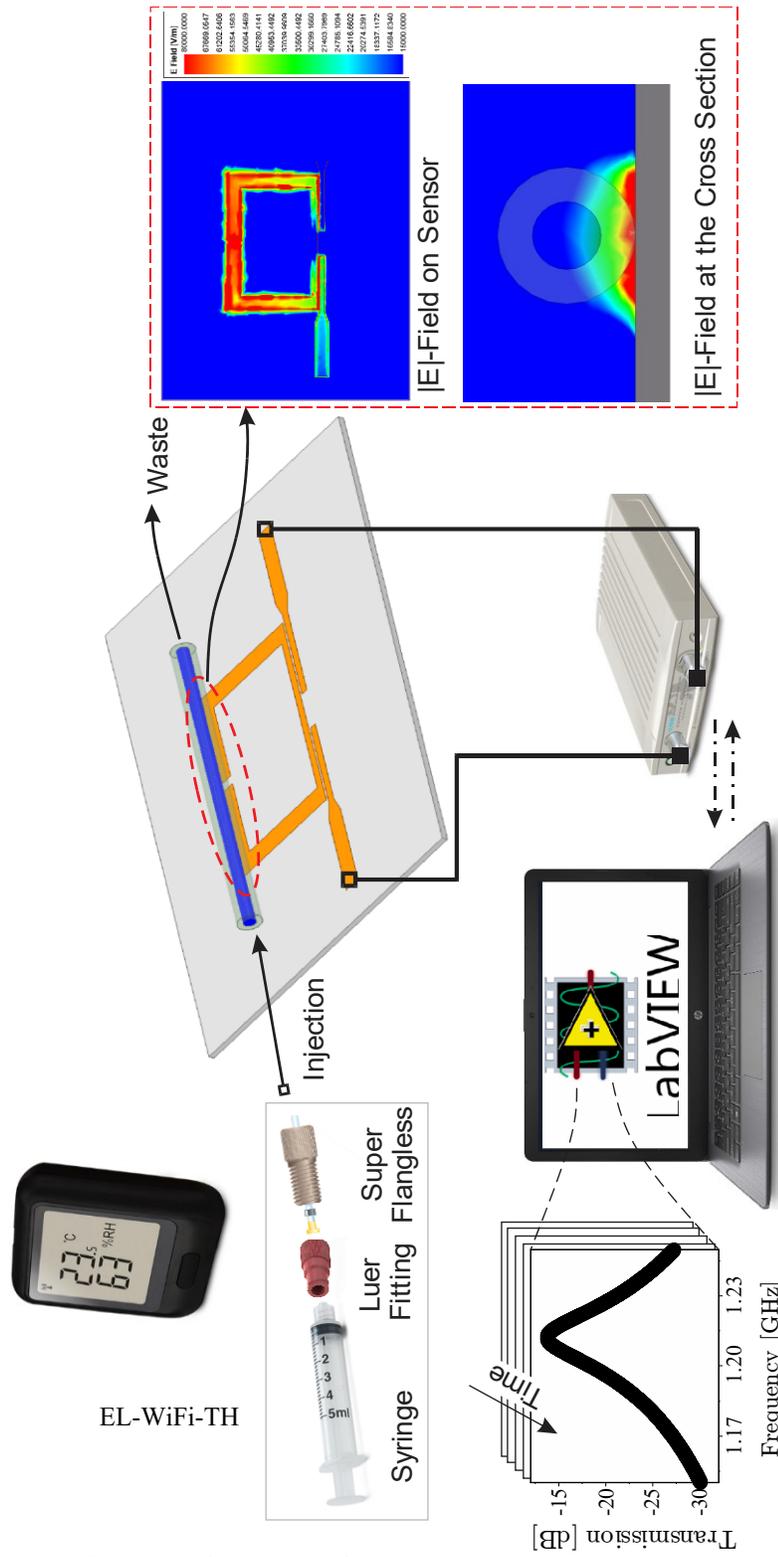


Figure 3.2 Schematic of microwave sensor and measurement setup

The design is simulated in Ansys Electronics (HFSS), where the sensor as a two-port network is stimulated with the input signal and the scattering matrix is computed based on the reflected (S_{11} / S_{22}) and transmitted signal (S_{21} / S_{12}) as follows:

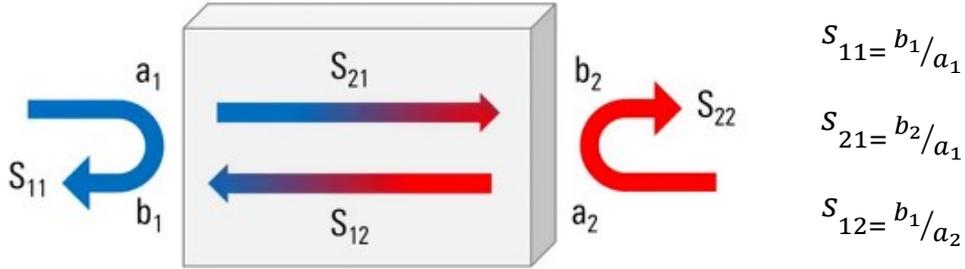


Figure 3.3 Scattering matrix in two-port network

Typically, the system is comprised of the sensor (see Figure 3.2) as the main two-port network with the following scattering parameters:

$$S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \quad 3.1$$

These parameters can be further simplified into the following due to symmetry ($S_{11}=S_{22}$) and reciprocity in nonlinear loss-less networks ($S_{21} = S_{12}$):

$$S = \begin{bmatrix} S_{11} & S_{21} \\ S_{21} & S_{11} \end{bmatrix} \quad 3.2$$

The reflection and transmission parameters of the proposed sensor are plotted below:

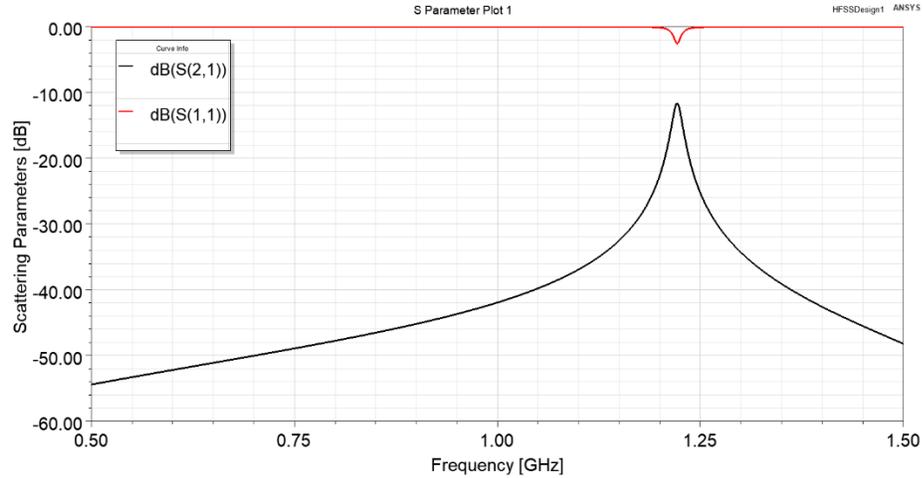


Figure 3.4 S-parameters of the sensor

The simulation result shows SRR holding sensitive regions to dielectric material (top-right section of Figure 3.2) where the warmer the color the higher its interaction with the material under test. The electric field is distributed unevenly, with a maximum value at the gap location and minimum at the opposite center of the SRR. This is a generic guide where to place material under test to have utmost interaction with the microwave field produced by the sensor. The tube on top of the resonator is placed right across the line of SRR that exhibits largest electric field concentration around itself. The cross-section of the tube shows how the field can penetrate into the environment, not far though. Therefore, the tube needs to sit as close to the sensor as possible.

The sensor is fabricated on Rogers 5880 substrate with dielectric properties of $\epsilon_r = 2.2$, $\tan \delta = 0.0009$. In order to measure the scattering parameters, S5065 Copper Mountain vector network analyzer is connected to the sensor with a pair of phase-compensated cables. This configuration is shown in Figure 3.2, with the peripheral details on the injection side. Since MUT is still in this experiment, it is injected inside the tube with conventional syringes. The relevant fittings for proper connection of the syringe and the PTFE tubing are shown in Figure 3.2. As long as the injected MUT covers the whole sensor, it is left untouched to undergo a temperature cycle. In the next section, the importance of temperature on the final results of the measurement is elaborated.

3.2 Problem Statement

Measuring dielectric properties of materials is a method of recognizing the variations in the environment matrix that is enabled with the direct accessibility of microwave power with the MUT. This non-contact feature of the planar sensors brings about disadvantages among easy accessibility to the desired MUT since its effect extends to consider all undesired impacts as well. This includes the effective variation in the relative humidity, proximity of external third-party elements, couplings, change in the effective dielectric constant of the MUT/substrate. This thesis is more oriented on the last item, which has an undeniable reason of temperature variations in the environment. The main idea arises from the fact that the permittivity of dielectric material depends on the temperature as follows [73]:

$$\frac{(\varepsilon - 1)(2\varepsilon + 1)}{9\varepsilon} = \frac{4\pi\rho N_A}{3M} \left(\alpha + \frac{\mu^2 g}{3k_b T} \right) \quad 3.3$$

where M is molecular weight, ρ is density, α is molecular polarizability, N_A is Avogadro's number, μ is dipole moment of the molecule, k is Boltzmann's constant, and g is a correlation factor that characterizes the relative orientation between neighboring molecules. This expression explains the inverse relationship between temperature and the dielectric constant of materials.

It is evident that this makes it more complicated for the sensor to characterize the materials (to which category they belong to) when their effective permittivity changes due to unpredictable and uncontrollable temperature variations in the environment. In order to demonstrate the importance and severity of this problem, a set of simulations are conducted in HFSS with the various dielectric constants for each material according to the ambient temperature. For this purpose, several concentrations of methanol and acetone in water are made, mixed with volumetric ratios of 20 %, 40 %, 60 %, and 80 % methanol/acetone in water. The resultant binary mixture has an effective permittivity between methanol/acetone and water according to the following expression [74]:

$$\varepsilon_c = \varepsilon_m \left(1 + \frac{3v_f(\varepsilon_f - \varepsilon_m)}{(1 - v_f)(\varepsilon_f - \varepsilon_m) + 3\varepsilon_m} \right) \quad 3.4$$

where ε_m denotes the host medium with volume fraction v_m filled with the second liquid of permittivity ε_f with volume fraction $v_f = 1 - v_m$. This expression is called Maxwell-Garnett equation [75]–[79] that relates the resultant permittivity to the constituent components. It is crucial to note that permittivity values of the input MUT of the previous equation ($\varepsilon_m, \varepsilon_f$) are frequency and temperature dependent and can be described by single Debye model for each material (water, methanol, and acetone) as follows [74]:

$$\varepsilon(\omega, T) = \varepsilon_\infty(T) + \frac{\varepsilon_0(T) - \varepsilon_\infty(T)}{1 - j\omega\tau(T)} \quad 3.5$$

where $\varepsilon_0(T), \varepsilon_\infty(T), \tau(T)$ are static (low frequency) permittivity, high frequency permittivity, and relaxation time, respectively, that are all temperature dependent inherently.

The goal here is to obtain the permittivity of water, methanol, and acetone plus binary mixtures of methanol-in-water and acetone-in-water at various temperatures. The idea is to obtain the temperature dependent permittivity values for bulk materials at the frequency of interest and use Maxwell-Garnett expression to deduce dielectric constants of the intermediate concentrations.

The input parameters of the materials under test, ($\varepsilon_0(T), \varepsilon_\infty(T), \tau(T)$) are found in the literature (water [80], acetone [81], and methanol [82]). The input parameters are given into single Debye expression to obtain the following characteristic graphs for each material:

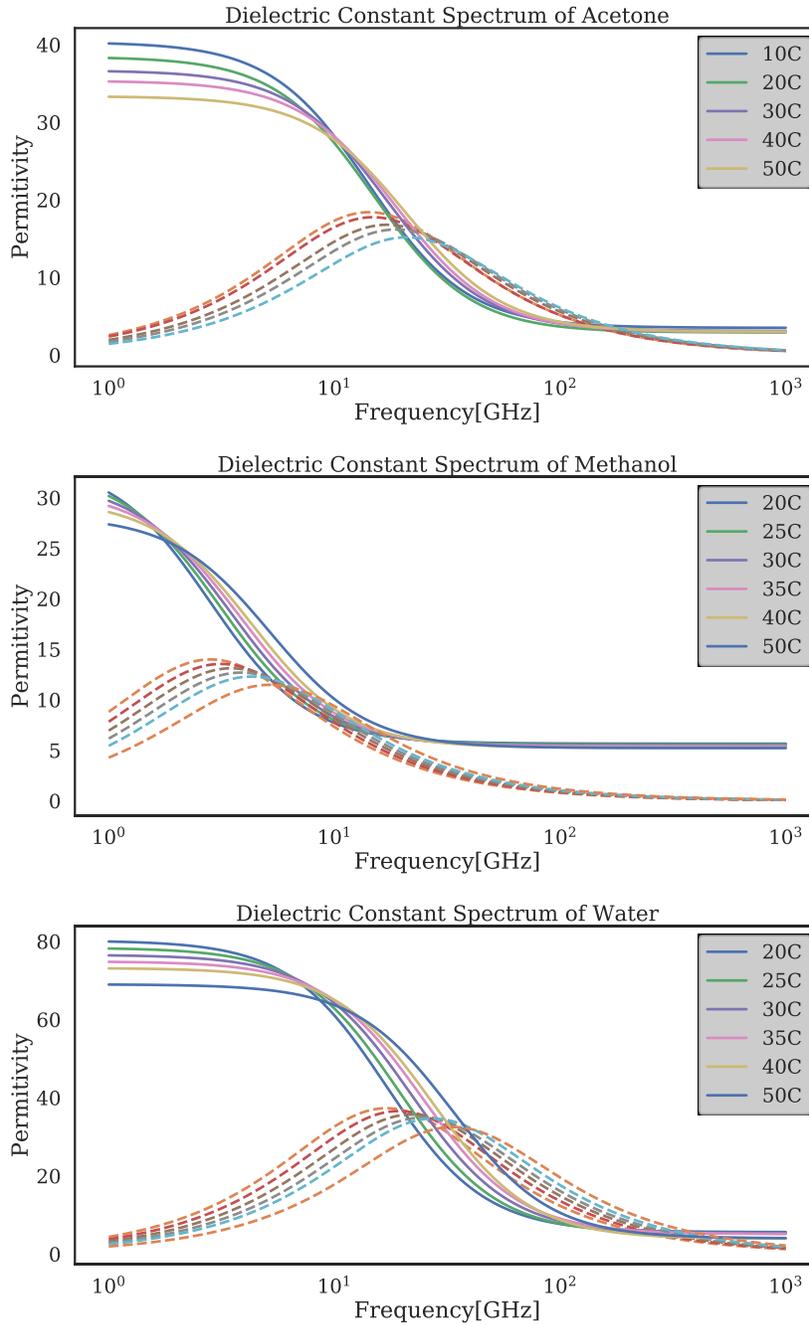


Figure 3.5 Dielectric spectrum of (a) Acetone, (b) Methanol, (c) Water at various temperatures based on single Debye model

The entire process is visualized in the following graph where the output of Debye model at the frequency of interest is rearranged differently in order to have a table consisting of permittivity values for each material at certain temperatures (middle graph in Figure 3.6). Then, Maxwell-Garnett is applied with respect to the

binary mixture ratios of interest and the final permittivity values for all used solutions are extracted even for mixtures of bulk medium.

The resultant table including all permittivity values for all materials are shown in Table 3.1. These values are given as the material properties of the liquid inside the tube when running HFSS simulations.

Table 3.1 Dielectric constant values for various materials at different temperatures

MUT	$\epsilon_r', \tan\delta$	20 °C	25 °C	30 °C	35 °C	40 °C	45 °C	50 °C
Water	ϵ_r'	69.8	71.4	73.0	74.7	76.4	78.1	79.9
	$\tan\delta$	0.028	0.031	0.034	0.038	0.042	0.048	0.054
Acetone (20%)	ϵ_r'	61.3	62.8	64.3	65.7	67.1	68.7	70.2
	$\tan\delta$	0.030	0.033	0.036	0.039	0.044	0.049	0.055
Acetone (40%)	ϵ_r'	53.5	54.8	56.2	57.4	58.6	59.9	61.3
	$\tan\delta$	0.032	0.036	0.039	0.042	0.046	0.051	0.057
Acetone (60%)	ϵ_r'	46.2	47.4	48.7	49.7	50.7	51.8	53.0
	$\tan\delta$	0.035	0.038	0.041	0.044	0.048	0.053	0.058
Acetone (80%)	ϵ_r'	39.5	40.6	41.7	42.5	43.39	44.4	45.4
	$\tan\delta$	0.039	0.042	0.044	0.047	0.050	0.055	0.060
Acetone (100%)	ϵ_r'	33.2	34.2	35.2	35.9	36.5	37.4	38.2
	$\tan\delta$	0.043	0.046	0.048	0.051	0.053	0.058	0.062
Methanol (20%)	ϵ_r'	59.7	61.0	62.4	63.9	65.2	66.7	68.1
	$\tan\delta$	0.044	0.049	0.054	0.060	0.067	0.075	0.080
Methanol (40%)	ϵ_r'	50.5	51.6	52.8	54.0	55.1	56.3	57.4
	$\tan\delta$	0.064	0.070	0.078	0.086	0.096	0.107	0.110
Methanol (60%)	ϵ_r'	42.1	43.0	44.0	45.0	45.9	46.8	47.6
	$\tan\delta$	0.087	0.096	0.106	0.117	0.130	0.145	0.146
Methanol (80%)	ϵ_r'	34.4	35.1	35.9	36.7	37.4	38.1	38.7
	$\tan\delta$	0.116	0.128	0.142	0.157	0.174	0.193	0.193
Methanol (100%)	ϵ_r'	27.3	27.9	28.5	29.1	29.6	30.1	30.5
	$\tan\delta$	0.156	0.190	0.211	0.211	0.233	0.259	0.256

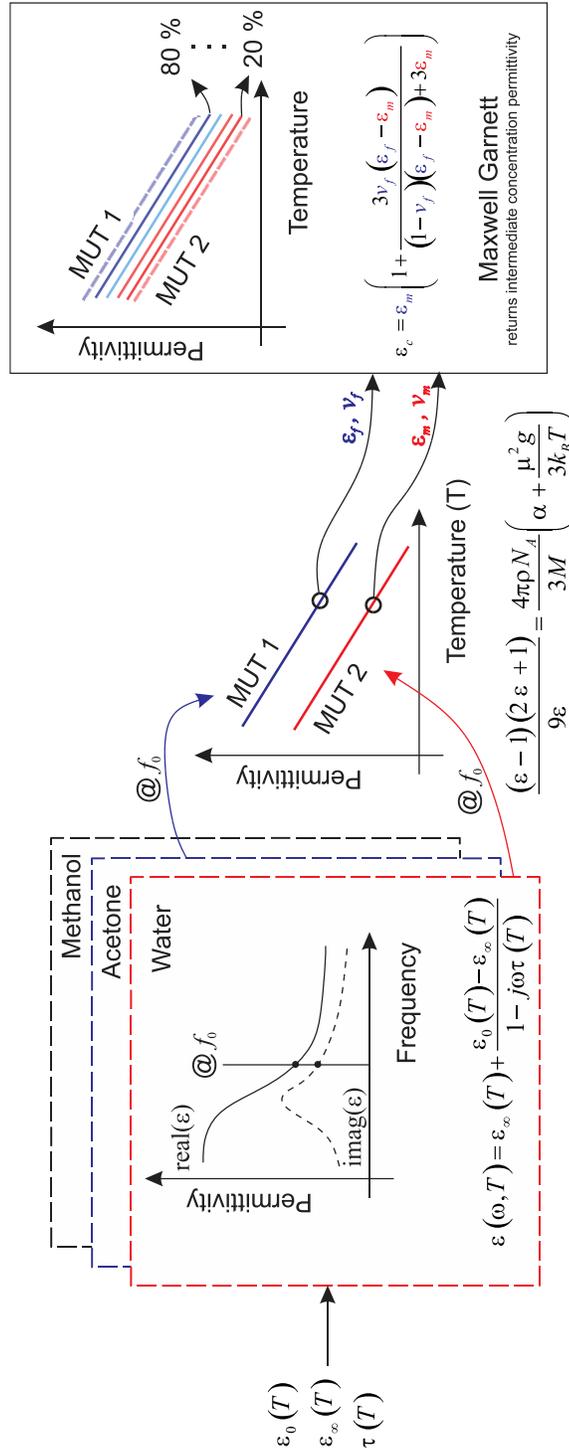


Figure 3.6 Flow graph for generation of dielectric constant values for various concentrations on materials in water at different temperatures

HFSS helps understanding the variations in the scattering parameters due to temperature. To this end, the material inside a PTFE tubing right on the sensor has dielectric constant properties of Table 3.1.

The simulation over all temperatures for all materials under test results in the complex and intertwined graph as shown below. This is an embodiment of what happens when different materials are exposed to temperature variation in the environment from room temperature $\sim 25^{\circ}\text{C} - 50^{\circ}\text{C}$. This adds ambiguity to the material characterization scheme due to the imposed deviation from temperature on the dielectric properties of either MUT or substrate. Assume that the sensor reports a frequency of resonance, this is the most frequently used parameter from conventional microwave sensors. With the change in the temperature, resonance frequency is impacted by the environmental variation as well as the amplitude and quality factor of the sensor. It is quite misleading to rely only on the reported resonance frequency.

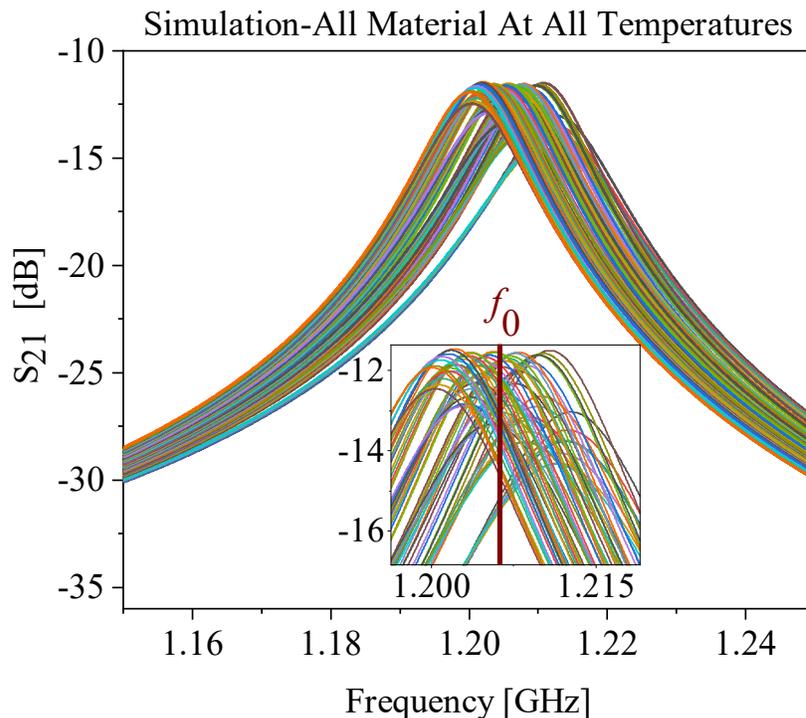


Figure 3.7 Simulating all materials inside tubing of sensor at various temperatures in HFSS, the inset focuses on a narrow bandwidth to demonstrate the complexity in selecting the correct graph with respect to the resonance frequency

3.3 MUT Preparation

In order to demonstrate this impact on the fabricated sensor, an experiment is conducted with acetone, methanol, and their binary mixtures in water. The required volume fraction of each material is taken with pipette to have high accuracy and combined to have the total volume of 5 ml. This mixture is shaken well for 20 seconds to achieve a decent homogeneity in solution. Then, the solution is injected inside the PTFE tubing using a syringe and proper fittings as shown in Figure 3.2. The liquid is left inside the tube for 5 minutes to let it rest and reach an equilibrium between the ambient temperature and the solution temperature inside tubing. Next, a temperature cycle is applied using a heater to the sealed chamber. Following graphs shows how the resonance frequency of the sensor changes in line with the temperature variation over two successive cycles, which confirms the repeatability of the sensor.

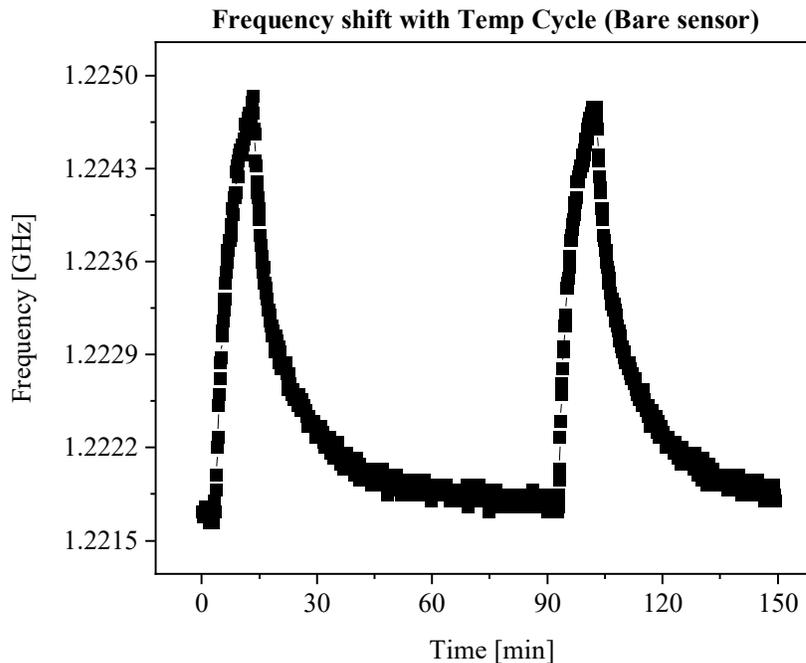


Figure 3.8 Repeatabile and reproducible temperature cycle in frequency of operation and its coincidence with the temperature

Along with the change in the bare sensor's characteristics, this trend is observed when the sensor is filled with various materials as well. First, acetone with volume fractions of 0.2, 0.4, 0.6, and 0.8 is mixed with water and experimentally verified the variations in each material as shown below. The first few minutes of the experiment is spent to stabilize the sensing platform with the environmental temperature for the purpose of consistency. Next, the heater is turned on and a commercial sensor inside the chamber demonstrating the temperature helps when to stop the heater. The high end of the temperature cycle is obtained when the commercial temperature sensor reads $\sim 50^{\circ}\text{C}$. This also helps consistency between different materials. Upon turning off the heat source and removing the lid off of the sealed chamber, the temperature drops naturally as a result of heat dissipation and the resonance frequency also decreases towards the original value of the one at the room temperature.

This temperature cycle is applied for methanol and its binary mixtures as well. It is instructive to learn that the resonance frequencies have huge overlap regions when analyzed exclusively. This is the main drawback for the planar sensors that in case of, say acetone, $f_0 = 1.195 \text{ GHz}$ can be attributed to either acetone 40%, acetone 20%, or water each at different ambient temperatures. The same issue exists for the methanol as well.

To make this problem more complicated, let us assume the sensor is assumed to measure wider variety of materials including both acetone and methanol plus their binary mixtures in water. In this case, the following graphs illustrates how intermingled frequencies are supposed to contain information about the properties of the materials inside the tube. This circumstance brings about more confusion when considering the same frequency of $f_0 = 1.195 \text{ GHz}$ to correlate with a single MUT whereas in this experiment it is evident that all water, methanol/acetone 20% and methanol/acetone 40% are among the candidates to exhibit the same resonance frequency at different temperatures. The real problem in industry lies where the temperature of the environment changes unpredictably and the sensor is expected to respond accurately and quickly.

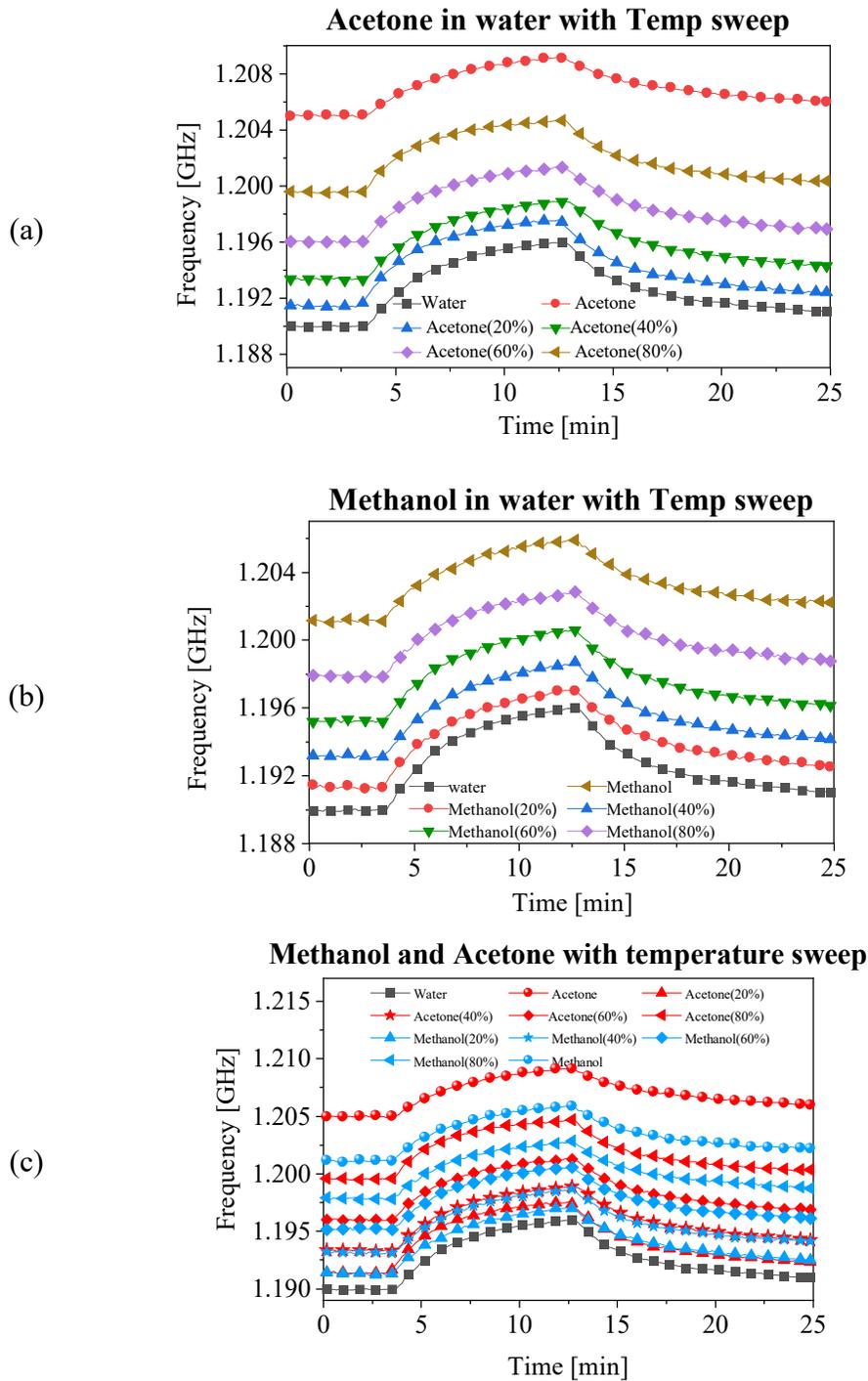


Figure 3.9 Frequency of resonance change in (a) acetone-water and (b) methanol-water solutions with volumetric ratio of 0 : 20% : 100 % over temperature cycle, (c) Mixture of the acetone-in-water and methanol-in-water

The measured results are post-processed to extract not only frequency of resonance, both also the quality factor and amplitude of resonance. The quality factor is a measure of sharpness of the bell-shaped resonance profiles and is defined as:

$$Q = \frac{f}{\Delta f_{3-dB}} \quad 3.6$$

Where f is the resonance frequency, and Δf_{3-dB} is the 3-dB bandwidth of the transmission profile as shown in the following graph.

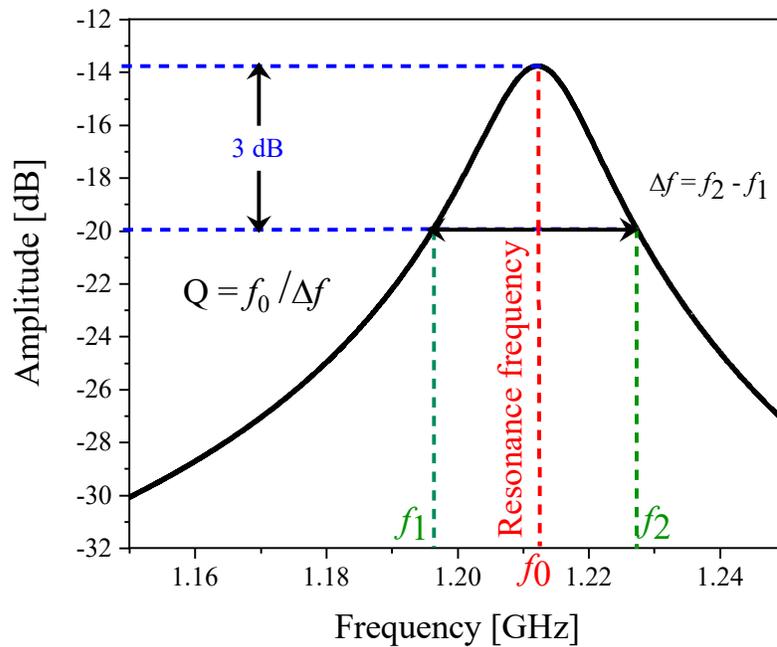


Figure 3.10 Quality factor definition based on resonance frequency and 3-dB bandwidth

The three components of each measured profile are sketched with respect to each other as shown below. It is evident that even these projection of the 3D graph composed of amplitude, frequency, and quality factor intersect at some temperatures, which elaborates how the measurements are mixed and can be confusing the a sensor considering an error bar on the data when the sensor is employed in industrial noisy environment.

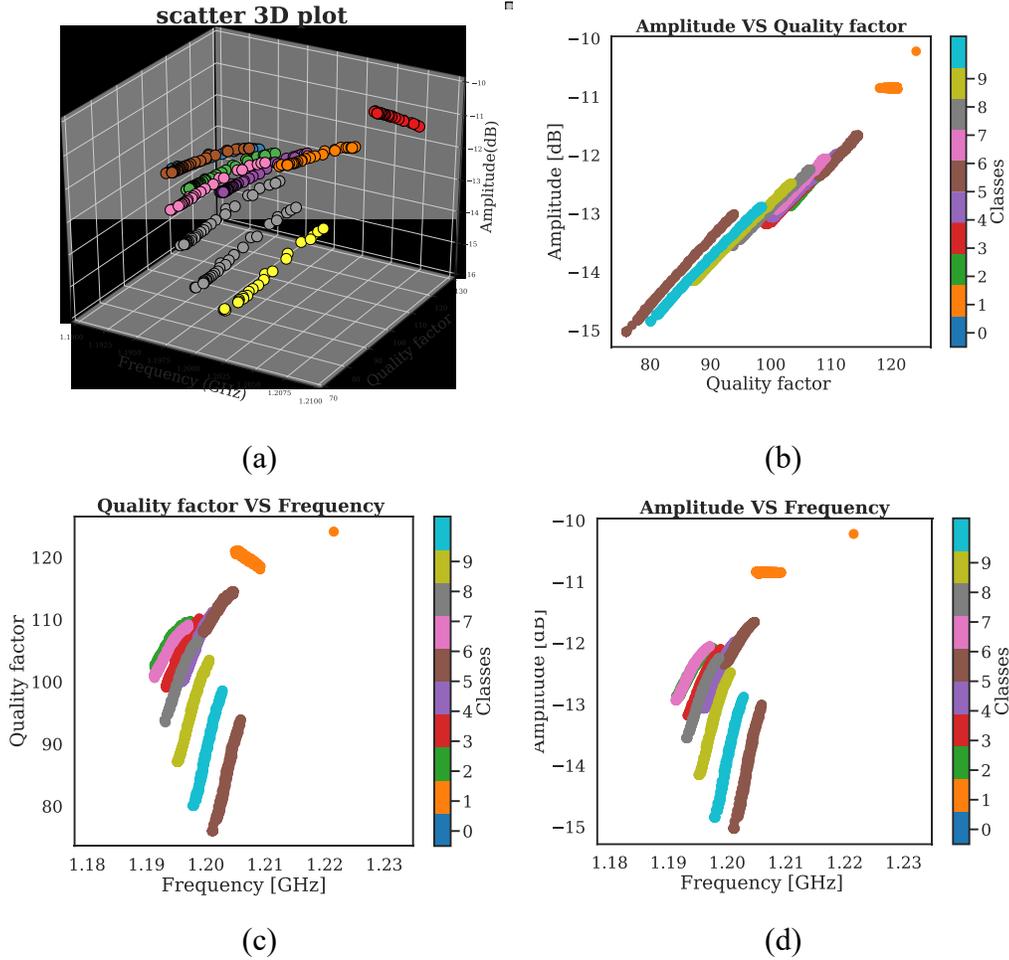


Figure 3.11 (a) 3D representation of recorded frequency, quality factor, and amplitude of various concentrations of methanol/acetone in water, which is projected in (b) amplitude vs. quality factor, (c) quality factor vs. frequency, and (d) amplitude vs. frequency

The simulation of the proposed design is performed in HFSS and evolutionary steps are taken to make sure the final result matches with that of the measurement. This process is quite informative from MW point of view, since the temperature of environment impacts on two major components of the sensor. First, the substrate contains a considerable capacitance part of the resonator, hence slightest change in its properties would result in the final resonance frequency quite extensively. The temperature dependency of the substrate is shown in [46] where its dielectric

constant reduces from 2.15 down to 2.1. Its impact is simulated only while the PTFE tubing holds the material under test, whereas no variation is applied on their permittivity value over the temperature cycle. Figure 3.12 shows how small is the contribution of the substrate and that it is not enough to match with the measurement. In the next step, the substrate is assumed to have no variation effect on the dielectric constant, but this time the dielectric constant of the materials is varied according to Table 3.1. Figure 3.12 demonstrates how much improvement we can gain considering the material variation that is considerably more than the substrate alone. Finally, the impact of temperature on dielectric constant is considered for both substrate and the material inside the tubing. Figure 3.12 showcases that this combination reveals the closest result to the actual measure values for the major bulk medium of water, acetone, and methanol.

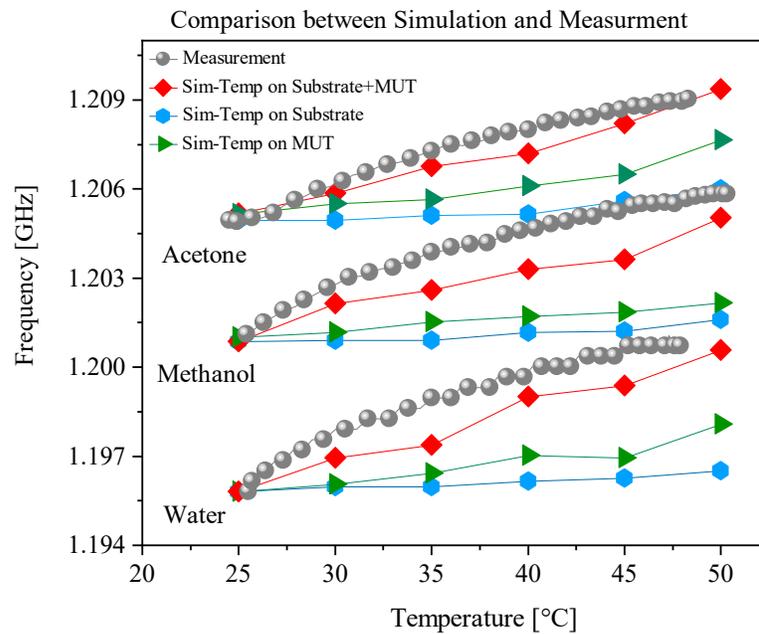


Figure 3.12 Temperature dependency of measured results is studied in simulation, considering the impact of substrate and MUT for water, methanol, and acetone.

Chapter 4

Data Preparation

To make a reasonable data set in terms of network input, the output data of the sensor should be post-processed. Here, the process is explained step by step as follows:

1. Recording the data as .s2p format
2. Extracting S_{21} to make classes
3. Making a huge matrix containing all the classes as a dataset
4. Feature extraction

An experiment is performed to acquire knowledge about the temperature effect on the microwave sensor. The sensor is connected to the VNA to monitor the scattering parameters. With the aid of LabView, which is an inevitable part of a time-based experiment, data is recorded every ten seconds as shown in Figure 3.2. Two types of data are recorded during the experiment, the temperature of the environment as well as S-parameters of the sensor. The combination of VNA and LabView records S-parameters, and the complementary commercial temperature logger (EL-WIFI-TH from EasyLog) located next to the SRR provides the temperature of the environment.

4.1 Data set preparation

4.1.1 Data as .s2p file

At the end of every experiment for each material, on average, 300 data points with S-parameters of the sensor are obtained. Frequency of interest combined with amplitude and phase of four scattering parameters (S_{11} , S_{21} , S_{12} , S_{22}) are embedded

in one s2p file (see Figure 4.1). Data points have s2p format, which consists of 9 columns and 5001 rows. There is no need to consider all nine columns of S-parameter as output data since S_{21} has enough information to train the network. In the following chapters, more details are provided to justify this claim.

	A	B	C	D	E	F	G	H	I
1	# Hz S DB R 50								
2	! CMT, S5065, 18077467, 19.2.3/2								
3	! Date: 3/28/2020 10:08:32 AM								
4	! Data: Format (Calibration Info)								
5	! Freq	S11:dB	S11:Ang	S21:dB	S21:Ang	S12:dB	S12:Ang	S22:dB	S22:Ang
6	1.10E+09	-9.16E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.61E+01	-1.05E+02	-7.97E-02	1.71E+02
7	1.10E+09	-9.17E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.67E+01	-1.08E+02	-8.06E-02	1.71E+02
8	1.10E+09	-9.16E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.62E+01	-1.07E+02	-9.05E-02	1.71E+02
9	1.10E+09	-9.11E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.59E+01	-1.06E+02	-8.04E-02	1.71E+02
10	1.10E+09	-9.07E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.62E+01	-1.07E+02	-9.63E-02	1.71E+02
11	1.10E+09	-9.12E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.61E+01	-1.04E+02	-6.65E-02	1.71E+02
12	1.10E+09	-9.05E-02	1.64E+02	-3.62E+01	-1.06E+02	-3.64E+01	-1.05E+02	-7.28E-02	1.71E+02

Figure 4.1 Snapshot of the scattering parameter for single temperature

4.1.2 Extracting S_{21} as dataset

Since the sensor under study is passive and symmetric, reflections from either port is the same, thus:

$$\begin{aligned} S_{11} &= S_{22} \\ S_{21} &= S_{12} \end{aligned} \quad 4.1$$

On the other hand, since the reflected power and transmitted power add up to 1, then one can only stick to S_{21} (transmission from port 1 to port 2) since S_{11} (reflection of inserted power from port 1) can be immediately generated using well-known microwave expression (with loss-less assumption of the system) as follows:

$$\begin{aligned} |S_{11}|^2 + |S_{21}|^2 &= 1 \\ S_{11}S_{12}^* + S_{21}S_{22}^* &= 0 \end{aligned} \quad 4.2$$

Therefore, logging S_{11} would only provide redundant information.

Among all parameters in s2p, S_{21} is selected as the output data of the sensor. Let us consider methanol as material under test (MUT). The temperature of the ambient changes from 22°C to 50°C; recording of these measurements yields 300 columns of S_{21} . These columns are transposed and concatenated to form a class

related to methanol. This very primitive dataset consisting of 300 rows and 5001 columns and one extra column which belongs to the type of material. In the proposed machine learning system, these columns represent classes of the learner (e.g. neural network).

4.1.3 Constructing a Mega-Matrix containing all the classes

Until here, each material has its own dataset. Different matrices, related to individual materials, are concatenated vertically to form a dataset containing all needed classes. In the end, a mega-matrix of data is obtained consisting of 3403 rows and 5002 columns that represent S_{21} profiles related to each material and its class. Another column is added next to the class column, which contains the corresponding temperature of each S_{21} profile, all shown in Figure 4.2.

	$S_{21(1)}$	$S_{21(2)}$	$S_{21(3)}$	$S_{21(4)}$	$S_{21(5000)}$	$S_{21(5001)}$	Class	Temperature(0C)
0	-72.492	-77.638	-75.736	-74.872	-73.07	-72.67	0	(24 ^{0C})
1	-78.157	-74.443	-82.462	-82.409	-72.528	-72.91	0	(24.5 ^{0C})
2	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
166	-73.575	-79.886	-76.105	-76.362	-73.458	-72.626	0	(50 ^{0C})
167	-73.248	-77.867	-78.244	-71.005	-73.399	-74.978	1	(24 ^{0C})
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1582	-76.838	-75.138	-78.823	-77.259	⋮	⋮	-72.817	-69.803	1	(50 ^{0C})
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1840	-73.635	-78.927	-80.38	-76.442	-73.331	-71.061	9	(24 ^{0C})
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3404	-73.864	-76.582	-73.65	-73.445	-69.92	-74.473	9	(50 ^{0C})

Figure 4.2 Scattering parameters stacked into single mega dataset including classes and temperature each recorded row belongs to

In addition, there is another method of preparing a dataset using electrical features of the sensor. Working with the s2p profile has its advantages and disadvantages. S2p contains all the information, including amplitude, resonance frequency, quality factor, and also the phase of the system. Thus, there is no missing information regarding the sensor's output. Moreover, feature extraction from the s2p profile takes a long time to make a dataset since one needs to do extra post-processing of the data. However, working with a massive amount of data can be considered a disadvantage for datasets based on s2p profile. In fact, there is a trade-off between speed and accuracy.

4.1.4 Feature Extraction

To have a dataset with a lower dimension and making the training process of the network faster, another type of dataset is introduced here.

The resonance frequency (F), amplitude (A), bandwidth (BW), and quality factor (Q) of a single S_{21} profile is shown in Figure 3.10. As it is mentioned earlier, for each material (let us consider methanol), around 300 data points are recorded. For each S_{21} profile, features (F, A, Q) should be extracted. The bandwidth's information is embedded in the quality factor. Thus, that is enough to consider these three columns (F, A, Q) as the feature of the profile. Now, 5001 data points of every S_{21} profile are summarized into three columns.

All in all, a matrix consisting of 300 rows (Figure 4.3), where each row has 5 columns, three for (F, A, Q), one for the corresponding class, and one for the temperature represents methanol's matrix. Matrices related to all materials are concatenated vertically to provide a dataset including 3404 rows and five columns containing all classes.

	A	B	C	D	E	F
1	F	Q	A	class	lable	Temp
2	1.22152	124.138	-10.2209		1 Acetone	26.6
3	1.20504	120.988	-10.8371		1 Acetone	26.6
4	1.20504	120.988	-10.8383		1 Acetone	26.7
5	1.205	120.984	-10.8382		1 Acetone	26.7
6	1.20508	120.992	-10.838		1 Acetone	26.8
7	1.205	120.984	-10.838		1 Acetone	26.9
8	1.20504	120.988	-10.8385		1 Acetone	26.9
9	1.20504	120.988	-10.8387		1 Acetone	26.9
10	1.20496	120.98	-10.8377		1 Acetone	26.9

Figure 4.3 Representation of stacked frequency, amplitude, and quality factor (F,A,Q) with respect to the classes, labels, and temperatures

4.2 Input of the Network

There are many factors that must be considered during network training, with input data playing the most important role. Among them, the following items have more importance in our current analysis:

4.2.1 Satisfactory S_{21} Profile

The sensor's output can be affected by environmental changes. These changes originate from the ambient and humans around the sensor. Temperature, pressure, and relative humidity variations can be considered environmental variations. On the other hand, human proximity to the sensor or displacing materials at different locations in front of the sensor are human errors. As errors are inevitable, it is necessary to find whether the affected output data is still meaningful or not. The following example supports this claim.

The SRR sensor is loaded with a series of water-methanol mixtures ranging from 0%-100% of methanol with a step of 10% increment. Measured transmission response of the SRR is recorded when the sensor is loaded with different concentrations of methanol while the temperature changes from 22°C to 60°C. The resonator in this study is a loss-compensated microwave sensor with tunable mode of operation with passive mode (no loss-compensation), active resonator mode (slightly loss-compensated), and oscillator mode (fully loss-compensated). The sensor under study is supposed to operate at active resonator mode, yet, because of compositional variations in the materials, it is possible for the sensor to experience transitions between modes of operation, which in any case is harmful to the sensing scheme and machine learning analysis.

The horizontal axis of S_{21} profiles are always fixed frequencies that the system is measured at. Because the frequencies are equidistant, the indices are sufficient to describe the relative relationship between each S_{21} profile. For each material, two hundred frequencies (points on horizontal axis) are chosen to be shown in each S_{21} profile for a better visibility (see Figure 4.4 for the extreme temperature on methanol and water). In this figure, it is evident that water causes the S_{21} profile shift downwards more due to its higher dielectric constant and the methanol has lower amplitude of transmission due to its higher loss, both at room temperature. As the temperature increases, both curves shift upwards with their quality factor recovered. Figure 4.5 demonstrates this for all classes involved: in each plot, as temperature increases, the S_{21} comes closer to the oscillation mode, which is undesirable. Quality factor and amplitude are two of the three features that are

important to us. In oscillation mode, there is no definition for quality factor since this is only a concept to describe the sharpness of a resonance profile. It is also well-known that amplitude remains relatively constant in oscillation mode since the output waveform of the transistor expands in amplitude until saturation regardless of environmental variations.

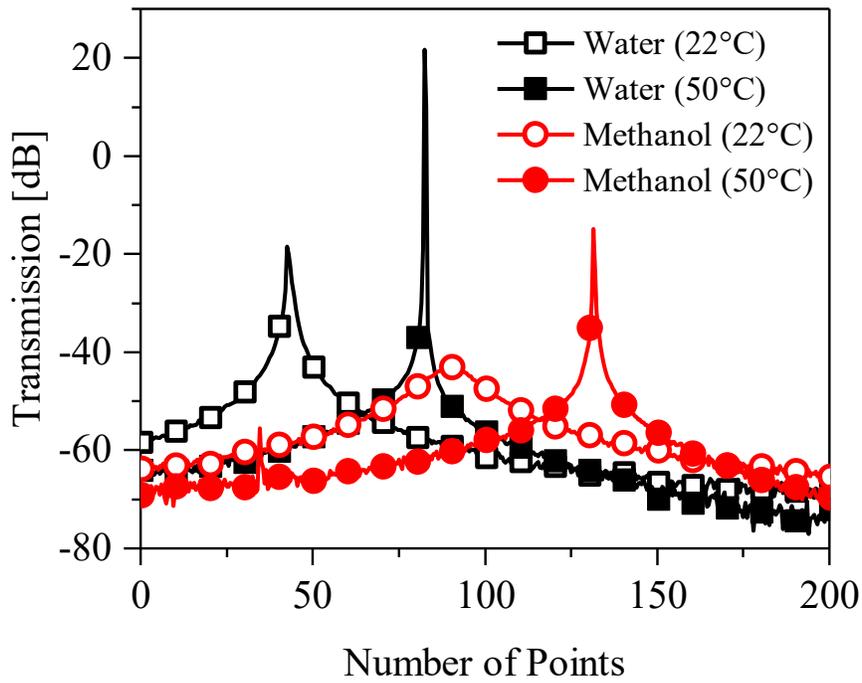


Figure 4.4 S_{21} for water and methanol at two extreme temperatures

Chapter 4: Data Preparation

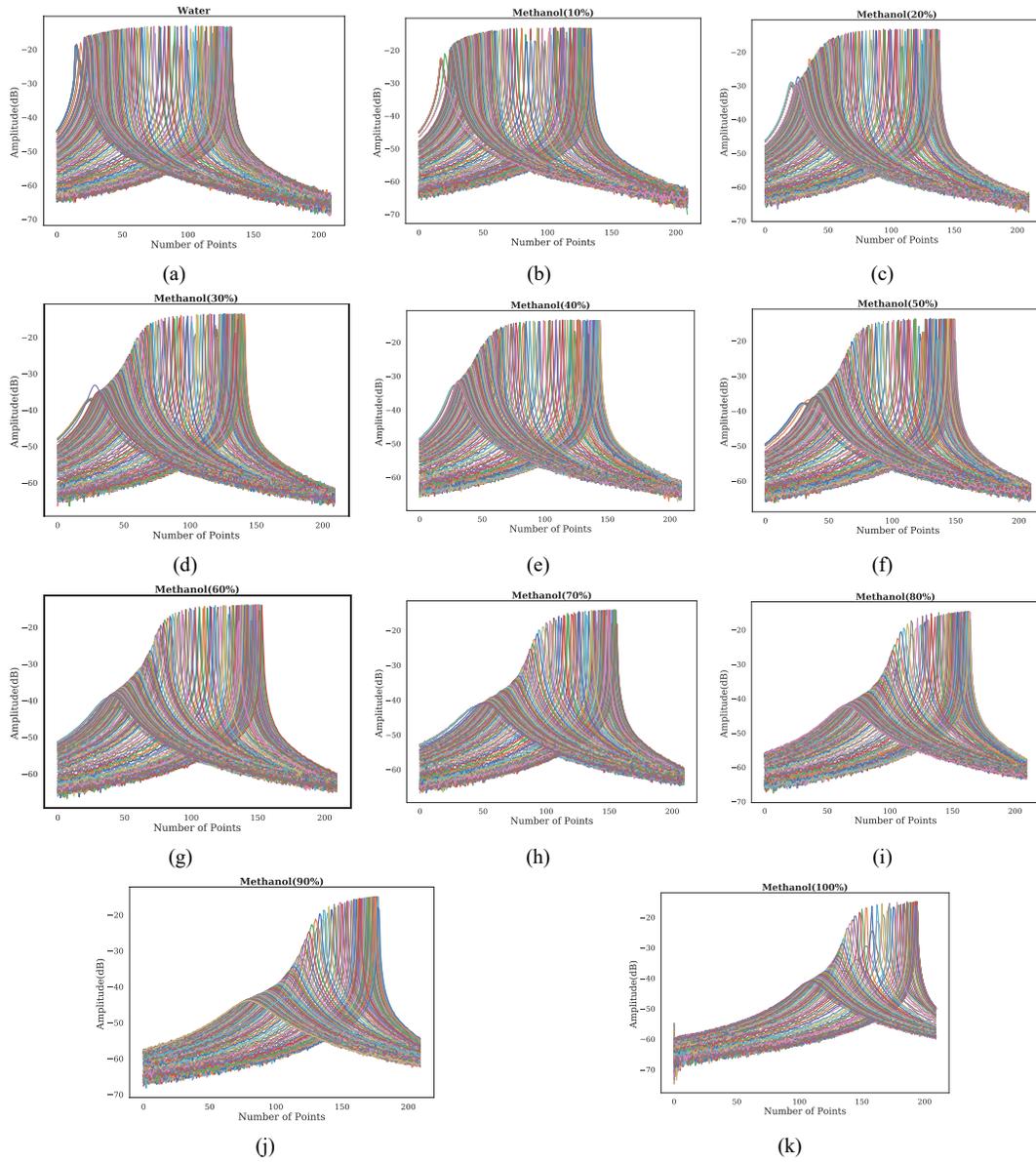


Figure 4.5 Various concentration of methanol in water with irregular and nonlinear sensor output

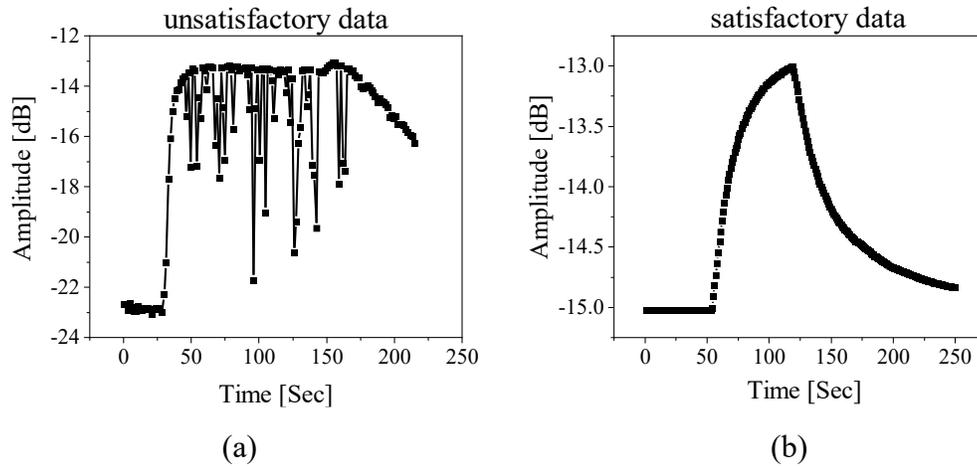


Figure 4.6 (a) Time-based description of defected amplitude performance in sensor, (b) Ideal amplitude response from the sensor

As the concentration of methanol increases, the S_{21} profile moves away from oscillation since methanol is more lossy than water (see Figure 4.5). If the network is fed with these types of data, the output result at the prediction stage would be irrelevant. The lower the concentration of methanol, the more mixed the data and the worse the predictions, as shown in Figure 4.7.

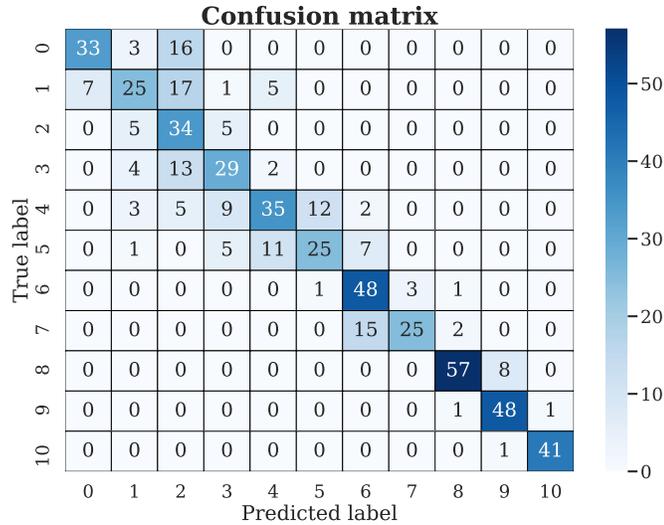


Figure 4.7 Confusion matrix for material characterization with poor sensor stability showing worse prediction at lower concentrations of methanol in water due to lower loss leading to instability in sensor

4.2.2 Connection between input and output

Inappropriate labeling, with labels (classes) incorrectly allocated to corresponding S_{21} profiles, may result in poor network training. If one or few input S_{21} profiles are mapped to incorrect classes, the network performance becomes poor due to incorrect recognition of the corresponding S_{21} profiles. It is necessary to check a few input samples that they are connected to the correct labels, and also repeat the same procedure after shuffling the data.

4.2.3 Shuffling the Dataset

In machine learning, not all data is introduced to the network at the same time. To describe the proper procedure, we need to introduce the following three terms: batch size, iterations, and epochs.

As mentioned earlier, the entire dataset is not passed through the network simultaneously. Rather, the dataset is split into batches. The number of training input data exemplars in each batch is defined as *batch size*.

Chapter 4: Data Preparation

An *epoch* refers to the cycle that an entire dataset is passed through the network with enough *iterations* to cover all batches, where the number of iterations is the quotient of training data size divided by the batch size.

To avoid similar data in each batch during network training, the dataset should be shuffled (otherwise, the learning process may be negatively affected). While shuffling the input data (i.e. presenting them in no particular order w.r.t label in the dataset), corresponding labels must be shuffled accordingly.

	s.1	s.2	s.3	s.4	s.5	s.6	s.7	s.8	s.9	...	s.4995	s.4996	s.4997	s.4998	s.4999	s.5000	classes	label	Temp
i.203	-36.190	-36.198	-36.210	-36.200	-36.194	-36.196	-36.170	-36.168	...	-38.770	-38.749	-38.798	-38.783	-38.787	-38.800		7	Methanol20%	44.4
i.630	-36.645	-36.636	-36.635	-36.634	-36.636	-36.619	-36.627	-36.613	...	-37.121	-37.129	-37.130	-37.126	-37.138	-37.143		1	Acetone	36.2
i.388	-36.402	-36.388	-36.386	-36.379	-36.375	-36.388	-36.362	-36.353	...	-38.703	-38.684	-38.704	-38.706	-38.713	-38.726		9	Methanol60%	19.7
i.426	-36.417	-36.429	-36.412	-36.418	-36.403	-36.408	-36.410	-36.394	...	-39.044	-39.029	-39.028	-39.041	-39.047	-39.043		3	Acetone40%	28.4
i.096	-36.096	-36.104	-36.096	-36.085	-36.078	-36.085	-36.068	-36.066	...	-39.178	-39.177	-39.166	-39.163	-39.176	-39.171		7	Methanol20%	30.3
i.470	-36.446	-36.443	-36.447	-36.453	-36.430	-36.433	-36.424	-36.418	...	-38.888	-38.912	-38.895	-38.884	-38.903	-38.905		3	Acetone40%	34.5
i.550	-36.517	-36.532	-36.539	-36.516	-36.516	-36.509	-36.506	-36.515	...	-38.596	-38.601	-38.590	-38.581	-38.590	-38.563		3	Acetone40%	45.0
i.705	-36.717	-36.698	-36.699	-36.713	-36.691	-36.681	-36.694	-36.666	...	-36.686	-36.691	-36.696	-36.718	-36.706	-36.721		1	Acetone	49.3
i.635	-36.620	-36.627	-36.634	-36.634	-36.616	-36.623	-36.613	-36.615	...	-38.546	-38.537	-38.559	-38.589	-38.551	-38.547		4	Acetone60%	38.3
i.605	-36.592	-36.592	-36.591	-36.594	-36.593	-36.583	-36.582	-36.574	...	-38.705	-38.727	-38.710	-38.723	-38.732	-38.727		4	Acetone60%	26.6
i.264	-36.243	-36.242	-36.239	-36.233	-36.251	-36.238	-36.217	-36.225	...	-38.898	-38.897	-38.915	-38.926	-38.920	-38.918		8	Methanol40%	32.4

Figure 4.8 Shuffled dataset

4.2.4 Imbalanced data

Another point in the dataset preparation stage is that the number of input data rows in each class should be comparable to that of other classes. In other words, if there is a significantly low/high number of input data for a specific class compared to the other classes, then the network can not sufficiently learn to discriminate the classes with lower input rows. For example, let us consider 2000 input rows for water and 100 for methanol. The ratio for training vs. total is 0.7, then the network has roughly 1400 rows for training water samples and only ~70 for methanol samples (please note that due to the random selection of training dataset from a shuffled total dataset, the numbers 1400 and 70 are approximate under uniform random selection). As shown in the following Figure 4.9, the number of data in each class should be similar to each other and in the same range. For training the network, the number of input rows should be sufficient for each class.

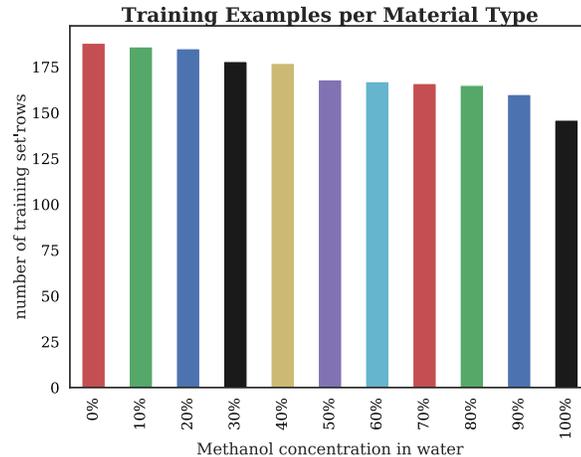


Figure 4.9 Distribution of measured data of various methanol concentrations in water

The following python code is to assign X_train and X_test with corresponding y_train and y_test values. Detailed information about this code is presented next.

```
1. X = dataset.iloc[:, 0:5000].values
2. y = dataset.iloc[:, 5003]
3. y_class = np.zeros((np.size(y),10))
4. for i in range(np.size(y)):
    y_class[i,y[i]] = 1
5. classes=["1","2","3","4","5","6","7","8","9","10"]
6. from sklearn.model_selection import train_test_split
7. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
    random_state = 1)
8. from sklearn.preprocessing import StandardScaler
9. sc = StandardScaler()
10. X_train = sc.fit_transform(X_train)
11. X_test = sc.transform(X_test)
12. y_train = np.asarray(y_train)
13. y_test = np.asarray(y_test)
```

Code 4.1 Python code for data preparation

4.2.5 Standard Scaler

It is important to ensure that the dataset is standardized to mean value of 0 and variance of 1 (line 9 of the code 4.1). Having the properties of standard normal

distribution is a crucial requirement for machine learning algorithm. It is presumed in the objective function of a learning algorithm that the features are centered around zero with a variance of one. The feature that has a larger variance with respect to other features might affect the objective function more and make the estimator unable to learn from other features correctly. In other words, with features being on different scales, certain weights may update faster than others since the feature values with a larger variance play a role in the weight updates.

4.2.6 Assigning X and y data

The dataset consists of input data points and output targets. Input data points include S_{21} profiles, with 5000 columns, in each row, which is shown with 'X' in code 4.1 and in the following analysis.

The network is expected to report material types as classes (with temperature compensation removing environmental errors) and the instantaneous material temperature. These values are regarded as two different output types denoted by 'y' in this analysis. Thus, the problem is divided into two separates, yet connected, stages. In the first stage, data is categorized into different classes with label assignment. For instance, water samples at different temperatures are all assigned to a single class label "Water," and similarly, all methanol samples are mapped to label "Methanol." In the second stage, since each row demonstrates the recorded data points at a specific temperature, each row is assigned to the corresponding temperature. Then, the network has two output types. The first includes text labels, while the second is comprised of numerical values.

The input data is divided into training and validation sets. The training set is used to train the network that is to map input data points to desired and defined output targets. In contrast, the validation set is an unseen set for the network from input data points that are used to evaluate the accuracy of the network after training.

4.3 Conclusion

This section is designated to how dataset is generated from sensor measurements using VNA. This is to help understand what the building

Chapter 4: Data Preparation

blocks of the system under study in terms of input data and their categorization into classes are. The section also describes possible discrepancies in measured data due to imperfection of the sensor design. It also considers the possibility of post processing data to reach a more concise format involving only frequency, amplitude, and quality factor rather than whole transmission parameters. Then, possible precautions are mentioned to apply on the dataset including shuffling, standardization, and balanced number of inputs. Finally, the recorded input data is divided into training and test sets.

Chapter 5

Machine Learning Algorithms to Characterize Materials with Varying Temperature

In this chapter, we describe that temperature cycles are expected to affect the sensor response that perturbs the correct classification of materials due to their cross-correlated dielectric constant patterns with respect to various temperatures. This problem becomes more challenging when the material under test is more than one and their signatures are intermingled when viewed at different temperatures. In this study, two materials of methanol and acetone are selected as a proof of concept to demonstrate the feasibility of the proposed machine learning system in classifying their various concentrations in water. It has already been shown how intertwined the scattering parameters of the sensor become when all concentrations of both methanol and acetone are exposed to a temperature cycle. However, in this section, several well-known algorithms including decision tree, K-nearest neighbors, and multilayer perceptron (neural network) are studied with the same given input dataset.

Upon performing classification, input materials are prepared with various concentrations of methanol/acetone in water within 0 – 100 % range, incremented by 20 %. This leaves us with 5 classes for methanol-in-water, 5 classes for acetone-in-water, all combined into 10 classes in total.

Each class of data holds temperature cycle on the materials inside the tube from 22°C up to 50°C. The first task for the machine learning system is to identify which class each data point belongs to. This is important to effectively remove the effect

of temperature and to recognize the material regardless of uncontrolled, variable environmental changes. It would also be interesting to elicit the temperature of operation at each data point, since this can be valuable information for applications in industry to know what happens at each temperature. Therefore, a methodology is devised to circumvent the effect of temperature, and to decipher its value at each measured data point for further post-processing. This chapter addresses both these challenges with machine learning algorithms.

5.1 Classification Algorithms

5.1.1 KNN

K-nearest-neighbors is a simple algorithm that uses the entire dataset in its training phase. Whenever a prediction is required for unseen data, it searches for K similar instances. The data with the most similar instance is finally returned as the prediction.

K denotes the number of nearest neighbors, which are used to vote the class of the new data (testing data). For example, if $K = 1$, then the testing data is given the same label as the closest known data point. Similarly, if $K = 3$, the labels of three closest classes are checked and the most common label is assigned to the testing data point. For illustration, let us consider two classes A and B, as shown in Figure 5.1.

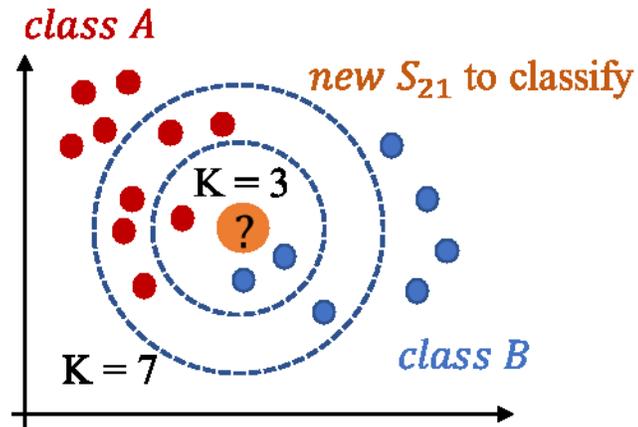


Figure 5.1 KNN prediction scheme

The new point (unseen data) is considered as a question mark in the graph. The KNN has to predict whether the new point belongs to class A or B. This process is also referred to as "voting." As mentioned earlier, K in the KNN algorithm refers to the number of nearest neighbors that to select. In Figure 5.1, there are two values for K . When $K = 3$, three points are chosen, which have the least distance to the new test point. Since the majority of points in the blue circle are blue (class B), the new data point is assigned to class B. Moving to higher number of neighbors, say $K = 7$, changes the prediction to class A as the majority of data neighborhood points belong to class A (red).

After splitting the data set into training and testing, the distance between every two data points should be calculated to check their similarity: the shorter the distance, the more similar the data points are. Then, K neighbors are selected such that they have the least distance from the new point. Once these neighbors are determined, the voting response for these K -neighbors is generated. Afterward, the algorithm determines to which classes the new points belong. Finally, the accuracy function provides more information on how accurate the algorithm is in its predictions. Few points should be considered to set the number of neighbors, K :

- For binary classifications, an odd value K must be chosen,
- When working with multi-class classification problems, the number of K must NOT be a multiple of the number of classes.

Program related to the KNN algorithm is shown in Code 5.1. The number of k (`n_neighbors`) sets to 7, which results in the best answer for the output. The confusion matrix, shown in Figure 5.2, confirms that all test data are assigned correct labels (classes). A minimum error is computed for different numbers of K, ranging from [1, 40]. The best K values should be chosen from 1 to 10. To obey the second rule in determining the value of K, which refrains us from setting this number equal to the number of classes, K = 10 has been removed.

```

1. knn = KNeighborsClassifier(n_neighbors=7)
2. knn.fit(X_train, y_train)
3. prediction = knn.predict(X_test)
4. X_new = X_test[:, 0 : 5001]
5. prediction = knn.predict(X_new)
6. error = []
7.
8. # Calculating error for K values between 1 and 40
9. for i in range(1, 40):
10.    knn = KNeighborsClassifier(n_neighbors=i)
11.    knn.fit(X_train, y_train)
12.    pred_i = knn.predict(X_test)
13.    error.append(np.mean(pred_i != y_test))
    
```

Code 5.1 Python code for KNN algorithm

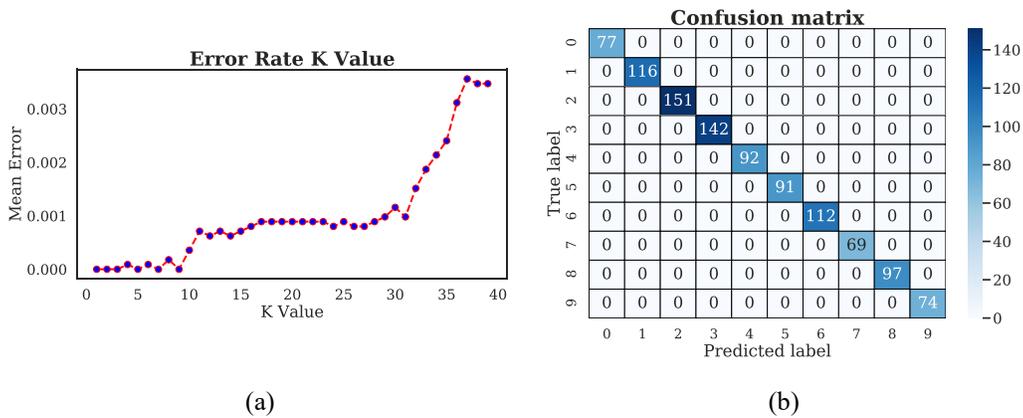


Figure 5.2 (a) Error rate for various K values in KNN, (b) Confusion matrix for K = 7

5.1.2 Decision Tree

The simplest method to perform classification is the decision tree algorithm. This algorithm is used for supervised learning, including two main types. One for

classification wherein the input data is categorical or discrete, and the other one for regression in that variable takes continuous values. In this study, the first algorithm is used to classify the types of material.

The outcome of the classification tree is Yes/No, which means the input data fed to the tree belongs to this class or not. Such a tree splits the data into partitions in each iteration through a process known as “binary recursive partitioning” to reach the desired output (here the type of material). In the decision tree algorithm, to reach the largest information gain, we start from the root and split the data based on the important features. In each iteration, data at each node is split until the samples at each leaf all belong to the same single class. This algorithm is easy to construct for simple datasets, and there is a high chance for overfitting. In the following, there is a simple algorithm to construct the model, fit the model using a train set and, finally, to predict the classes using "model.predict". The score from confusion matrix (see Figure 5.3) is about 0.992, which is very good and reasonable in this case.

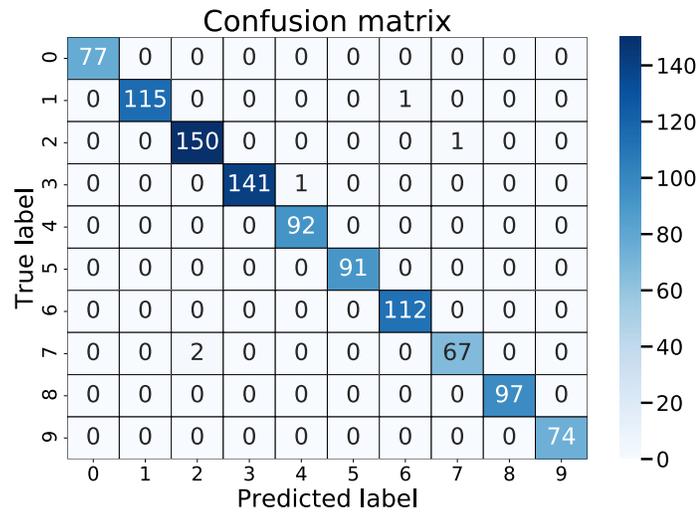


Figure 5.3 Confusion Matrix for Decision Tree

5.1.3 MLP

A multilayer perceptron (MLP) is a standard neural network model that combines many simple perceptrons to form a larger, more powerful network

[83], [84]. The following Figure 5.4 shows a simple example that represents a small MLP with an input layer, an output layer, and one hidden layer. There can be multi-hidden layers in MLP, but this section of the network is hidden from inputs/outputs. Neural networks are usually considered black boxes. This means that we provide input and output vectors and do not expect to fully understand what is going on in the potentially many hidden layers in between. In this section, though, those computations are analyzed to understand them.

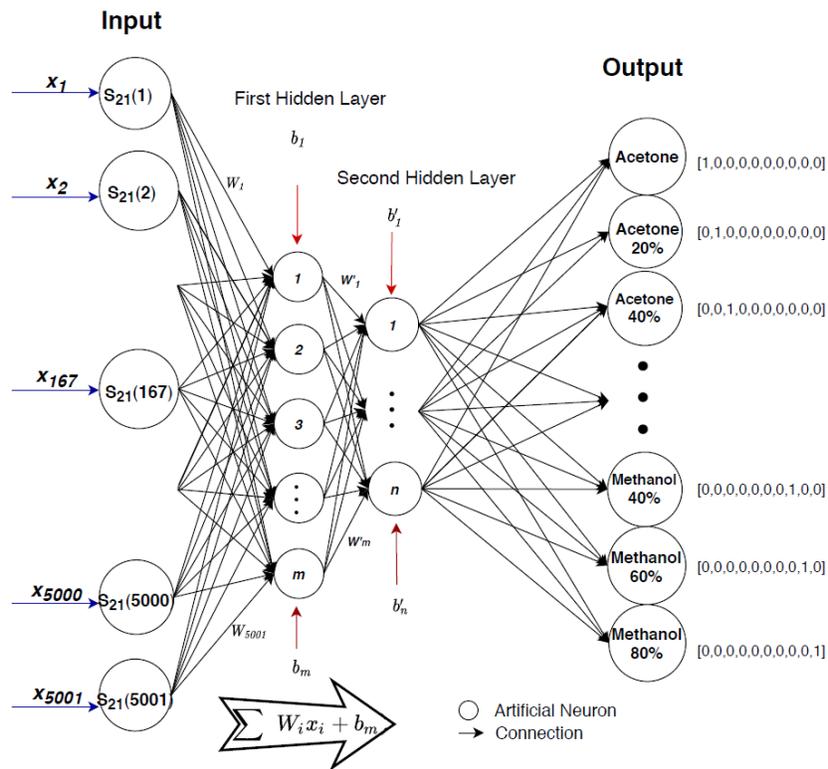


Figure 5.4 Simplified MLP structure

Let us consider input signals $x_1, x_2, \text{ and } x_3, \dots$ that are transmitted to the next (hidden) layer via the links, as shown in Figure 5.4. The neuron values are calculated using a summation across the inputs and the activation function. Here is the formula for how these hidden layer neurons are calculated:

$$h_n = \sigma \left(b_m + \sum_{i=1}^3 (x_i * w_{i,m}) \right) \quad 5.1$$

where σ , w , and b denote activation function, weight, and bias, respectively. The input values are multiplied by the corresponding weight values and summed up. Then bias value, one for each node, is added to the sum. Each neuron in the hidden layer has its own bias value, b_m . Subsequently, the activation function is applied to the latest sum. In general, there are different types of activation functions that can be applied to the network. Activation functions are also used with simple perceptrons, but that alone does not make them the powerful classification tools as multilayer perceptrons. However, it is vital to have more complex activation functions to accomplish difficult tasks such as classification and regression. Before moving on to demonstrate how the MLP algorithm is made, please note that the output layer neurons are treated similar to the hidden layer neurons in terms of biases and weights.

The MLP algorithm is described in the following, and its essential parts are explained. After splitting the input data to training and testing, to initialize the network, the classifier (model) should be defined. To clarify the difference between an "algorithm" and a "model," we provide the following definitions.

An "*algorithm*" in machine learning is a procedure that runs on the input data to create a machine learning "*model*." A "*model*", on the other hand, is the output of a machine learning algorithm. Their relationship can be defined as follows:

$$\text{Machine Learning Model} = \text{Model Data} + \text{Prediction Algorithm}$$

Here, we are after a machine learning "*model*," and the "*algorithm*" is only the path to follow to obtain the model.

In Keras (Python deep learning API), models are defined as a sequence of layers. Therefore, at the first step, the sequential model is established to add layers one at a time and build the structure of the network. The dimension of the input data is given to the network using `input_dim = 5001` to avoid making a mistake in the first layer. Other internal parameters inside the code are explained in detail in

Chapter 6. Thus, here, the emphasis is on network building. In this study, a fully connected network with three layers is defined. Two hidden layers, consisting of 100 and 40, respectively, and one output layer with ten neurons are added to the network using `classifier.add(Dense)`. The parameter "Dense" is used since the layers are fully connected.

Now, the model is defined, and it is time to compile it by calling `compile()` function. The compiler helps the model choose the best way for training and making predictions to run on the hardware (CPU, GPU) using efficient numerical libraries under a backend, such as TensorFlow.

Now the model is defined and, after compilation, it is ready to train or "fit" on the loaded data by calling `fit()` on the model. `X_train` and `y_train` are also passed to the fit function as arguments, as are "epochs" and "batch size." The number of epochs determines how many times the entire dataset is presented to the network; the batch size is a hyperparameter that defines the number of input data to introduce to the network in each iteration.

To evaluate the performance of the network, function `evaluate()` is used after the model is fitted. This function returns a list with two values; the first one is the loss, and the second one is the accuracy of the model on the dataset. Since the accuracy is more important, the second value of this list `accuracy[1]` is chosen to print.

```
1. # Initialising the ANN
2. classifier = Sequential()
3. # Adding the input layer and the first hidden layer
4. classifier.add(Dense(units = 100, kernel_initializer = 'uniform', activation = 'relu', input_dim
   = 5001))
5. # Adding the second hidden layer
6. classifier.add(Dense(units = 40, kernel_initializer = 'uniform', activation = 'relu'))
7. # Adding the output layer
8. classifier.add(Dense(units = 10, kernel_initializer = 'uniform', activation = 'softmax'))
9. # Compiling the ANN
10. classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
11. # Fitting the ANN to the Training set
12. classifier.fit(X_train, y_train, batch_size = 10, epochs = 30)
13. accuracy = classifier.evaluate(x=X_test, y=y_test)
14. y_pred = np.zeros_like(y_test)
15. y_pred_class = classifier.predict_classes(x=X_test)
16. for i in range(len(y_pred_class)):
17.     y_pred[i][y_pred_class[i]] = 1
18. print("The prediction Accuracy is : ", accuracy[1])
```

Code 5.2 Python code for MLP

predict() function can be used to predict to which classes unseen data belongs. However, for simplicity and to predict classes directly, *predict_classes()* is used here. The piece of code related to the structure of the network is given in Code 5.2.

The confusion matrix obtained for the network shows optimum prediction results with 100% accuracy in the prediction phase. Figure 5.5 shows that, after 13 epochs, the loss function converges well enough to provide the maximum accuracy.

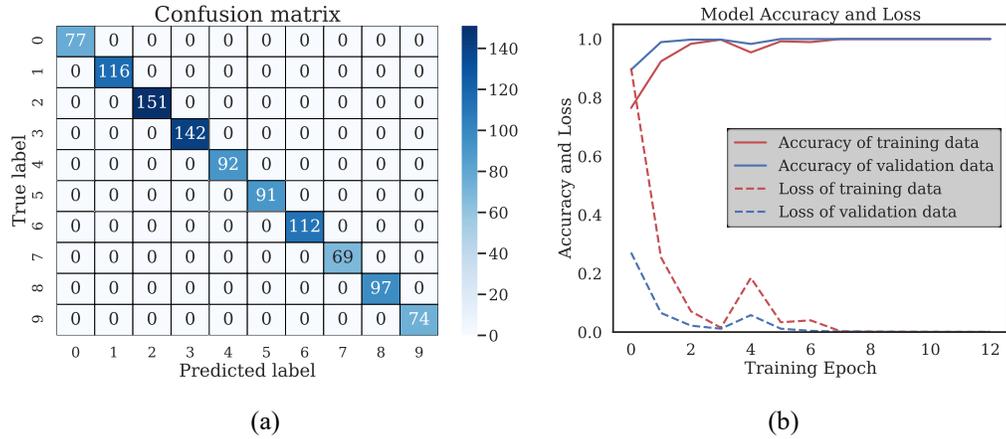
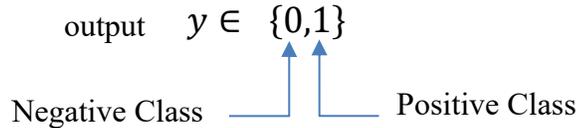


Figure 5.5 (a) Confusion matrix for MLP, (b) Convergence curve for MLP

5.1.4 Logistic Regression & PCA

The logistic regression algorithm is mainly used for binary classification, wherein one wants to know whether the unseen data belongs to a class or not (just like a yes/no question). For every class in the dataset, there is a scatter plot similar to Figure 3.11 based on S_{21} profiles. If a specific S_{21} belongs to the desired class, it takes the value of one. Otherwise, it maps to zero.



To develop a classification-algorithm, first, we need to fit a line to the data. After that boundaries are set such that the best fitting curve is achieved. One can assume that if the classifier output is greater than or equal to 0.5, it takes $y=1$, and conversely if it is less than 0.5, it takes $y=0$. Therefore, everything to the right of the middle point (0.5) ends up being predicted as the positive class and vice versa, i.e.

Threshold classifier $h_{\theta}(x)$ output at 0.5:

If $h_{\theta}(x) \geq 0.5$, predict “ $y = 1$ ”

If $h_{\theta}(x) \leq 0.5$, predict “ $y = 0$ ”

But there is a problem, if a data point far from other data points is added to the training step: the slope of the predictive line (see Figure 5.6(a)) rotates such that the one values are considered as zero which causes problems (see Figure 5.6(b)). That is why linear regression is not considered as a classifier.

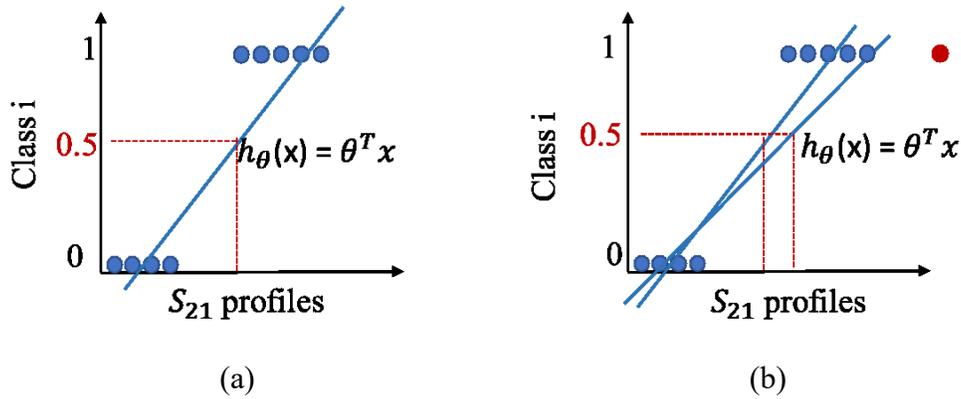


Figure 5.6 (a) Normal classification line for linear regression, (b) Deviation in predictive line of linear regression due to additional outlier

Thus, we develop a hypothesis that satisfies the following condition:

$$0 \leq h_{\theta}(x) \leq 1 \quad 5.2$$

To do so, the form of the linear function is modified to sigmoid or logistic function, and that is why the term logistic function is changed into the name *logistic regression*. Sigmoid function, as shown in Figure 5.7, asymptotes at one and zero as x goes to $+\infty$ and $-\infty$ respectively. Finally, given this hypothetical representation, the parameter θ must be fitted to our data.

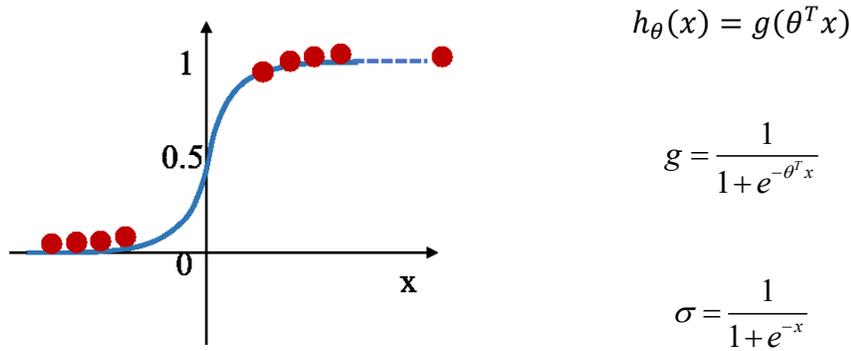


Figure 5.7 Sigmoid function with expressions

Here, the logistic regression is used for multi-class classification. Similar to binary classification using an idea called *one vs. other classification*, we can incorporate it in multi-class classification as well. More often, there is a learning problem due to having too many of features. These problems, including overfitting and underfitting, might occur while running different algorithms. Thus, it may be necessary to reduce the number of features. To do that, one can manually select which features to keep or use principle component analysis (PCA). In the case of microwave sensors, three features of the S_{21} profile that should be considered are resonance frequency, amplitude, and quality factor. In this study, as explained in Chapter 4, another dataset made of these features is considered too. It is important to mention that by removing some features, some information is missing, which is undesired. The variability of PCA is set to 95%, which means that the algorithm reduces the features such that 95% of the information is still available in the transformed data. The code related to the logistic algorithm follows.

```

1.  pca = PCA(.95)
2.  pca.fit(X_train)
3.  pca.n_components_
4.  X_train = pca.transform(X_train)
5.  X_test = pca.transform(X_test)
6.  logisticRegr = LogisticRegression(solver = 'lbfgs')
7.  logisticRegr.fit(X_train, y_train)
8.  score = logisticRegr.score(X_test, y_test)

```

Code 5.3 Python code for PCA classification

5.1.5 Comparative analysis

In this chapter, we describe results of several algorithms that have been employed to classify the sensor data influenced by temperature information. An overview of the final accuracy of each approach is provided in **Table 5.1**, where MLP has the best performance. Decision tree with 99.2 % and KNN with 99.9% also performed well. However, logistic regression achieved only 89.3 % accuracy and, as such, it has not been further considered in this study. In the following section, another regression algorithm is used. However, rather than as a classifier, it is used as a regressor to map the input data to a continuous output of temperature.

Table 5.1 Comparison between different algorithms

Algorithm	Decision Tree	KNN	MLP	LR+PCA
Score	0.992	0.999	1.00	0.893

5.2 Linear Regression Algorithm

Linear regression (LR) is one of the most straightforward algorithms in machine learning. It is a statistical model representing a relationship between two

variables using a linear equation. It involves graphing a line over a set of data points that most closely fits the overall shape of the data. The significant uses of regression analysis are in forecasting an effect, wherein the regression algorithm helps to understand how much the dependent variable changes with the change of one or more independent variables. In simple linear regression ($y = mX+c$), we are trying to find the correlation between X (as the input) and y (as the target/output) variable. It means every value of X has a corresponding value of y in the model. Thus, in contrast with classification algorithms, the output data in this algorithm is continuous (here, every S_{21} profile is assigned to a specific temperature). The data is modeled using a straight line. To optimize the regression model, each data point plays an important role. In this section, to predict the temperature for each material, a straight line is set to fit the input data. As is shown in Figure 5.8, the slope of the line is sensitive to the distribution of data. Two features that make this algorithm useful in many applications are:

- it is computationally inexpensive
- it is easily comprehended and can be represented using simple mathematical notation

To understand LR better, the algorithm is explained with respect to our needs. Here the x - and y -axis represents the actual and predicted values of temperature, respectively. First, a line is passed through the data; second, the distance from the line to the data is calculated. Third, each residual (the distance from a line to a data point) gets squared, and finally, they are added up. The same procedure is conducted for many lines with different slopes, as shown in Figure 5.8.

A plot with the sum of squared residuals on the y -axis and different lines on the x -axis is achieved after sweeping the slopes for a number of lines. Based on the outcome, the best line to fit the data has the least sum of squares. Least sum of squares estimates two parameters: y -axis intercept and slope. Each class has its fitting line. Figure 5.8 helps to understand that how much the predicted values are suitable for given problem.

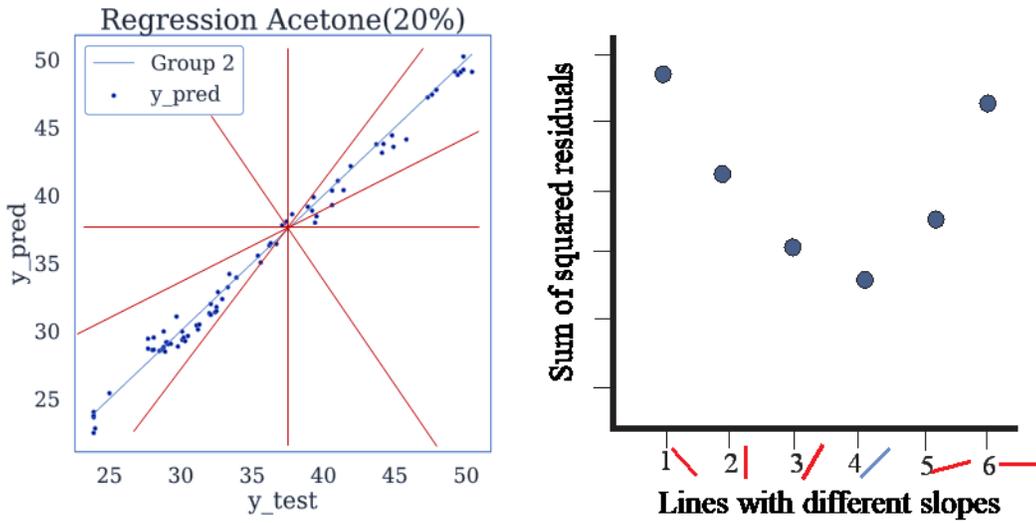


Figure 5.8 Fitting diagram for linear regression

The temperature data, shown in Figure 5.9, is read from a commercial temperature sensor (EL-Wifi-TH [85]) as located next to the sensor (see Figure 3.2). As mentioned in the data preparation section, besides the type of material (class), another column is considered in the dataset as a material temperature. Therefore, for each class, the temperature is increased continuously from 22°C to 50°C.

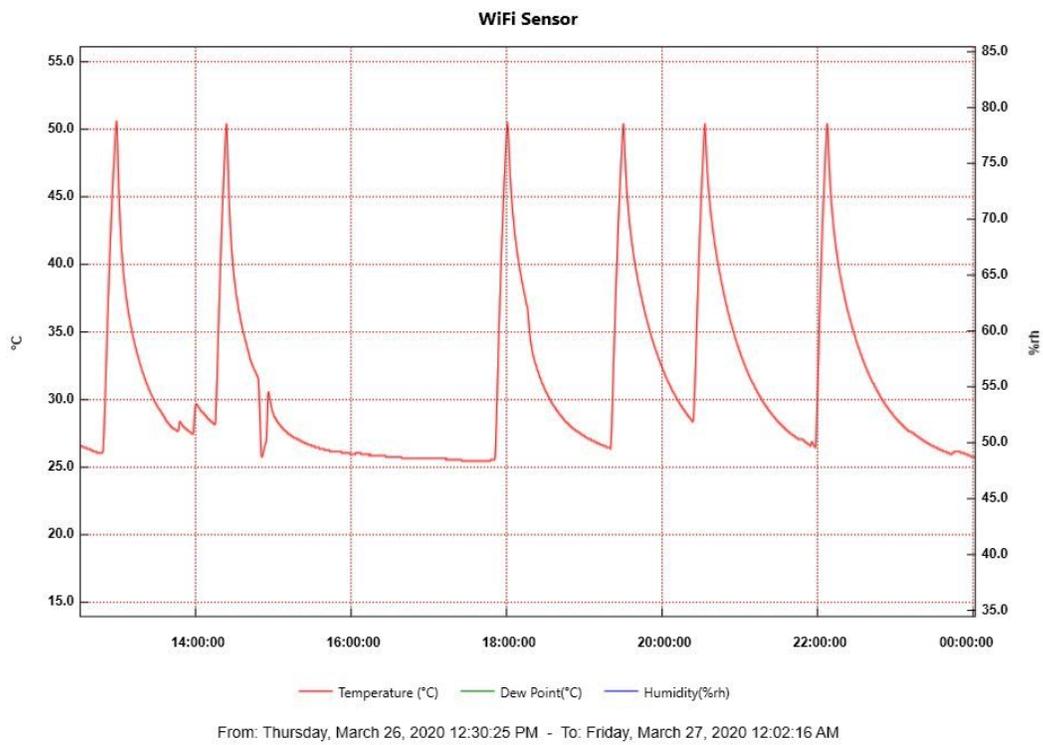
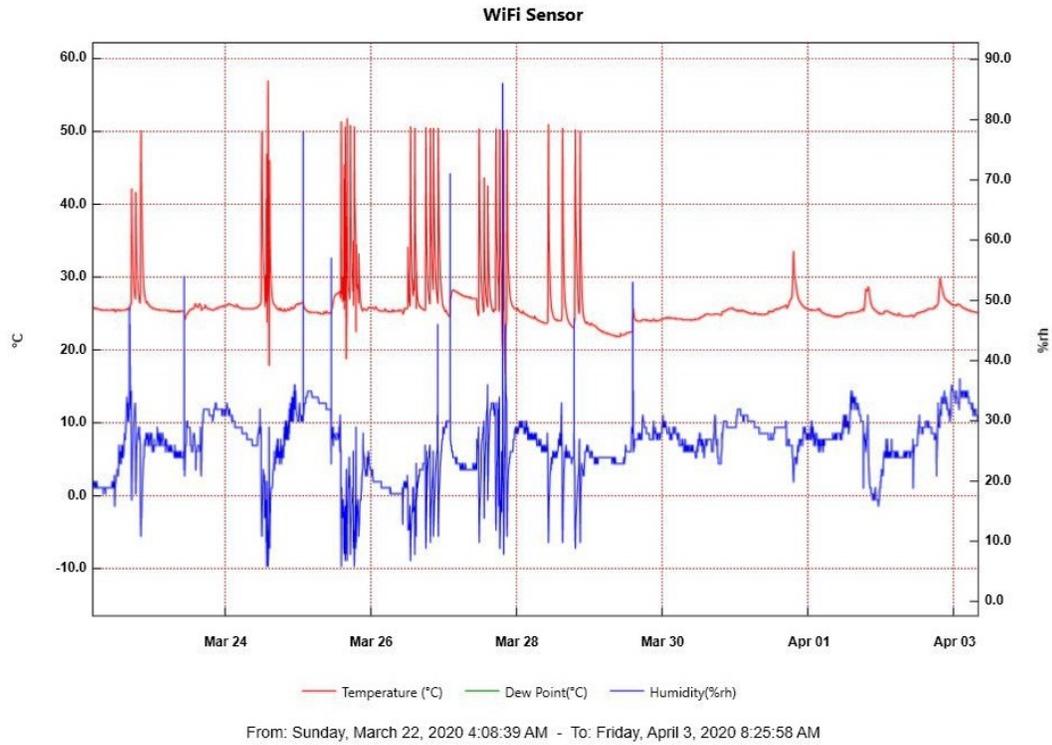


Figure 5.9 Temperature cycle during experiments

Chapter 5: ML Algorithms to Characterize Materials with Varying Temperature

The following code uses linear regression as the core of the algorithm. Data is passed through the “fit” and “predict” functions to provide the temperature of unseen S_{21} profile.

```
1. X1 = dataset.iloc[0:238,0:5000].values
2. y1 = dataset.iloc[0:238, 5003]
3. y1 = y1.values.reshape(-1,1)
4. X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.2,
    random_state=0)
5. regressor1 = LinearRegression()
6. regressor1.fit(X_train1, y_train1) #training the algorithm
7. y_pred1 = regressor1.predict(X_test1)
```

Code 5.4 Python code for linear regression, limited to only one class

In the following figure, the result of the linear regression algorithm for all classes is shown, see Figure 5.10.

Chapter 5: ML Algorithms to Characterize Materials with Varying Temperature

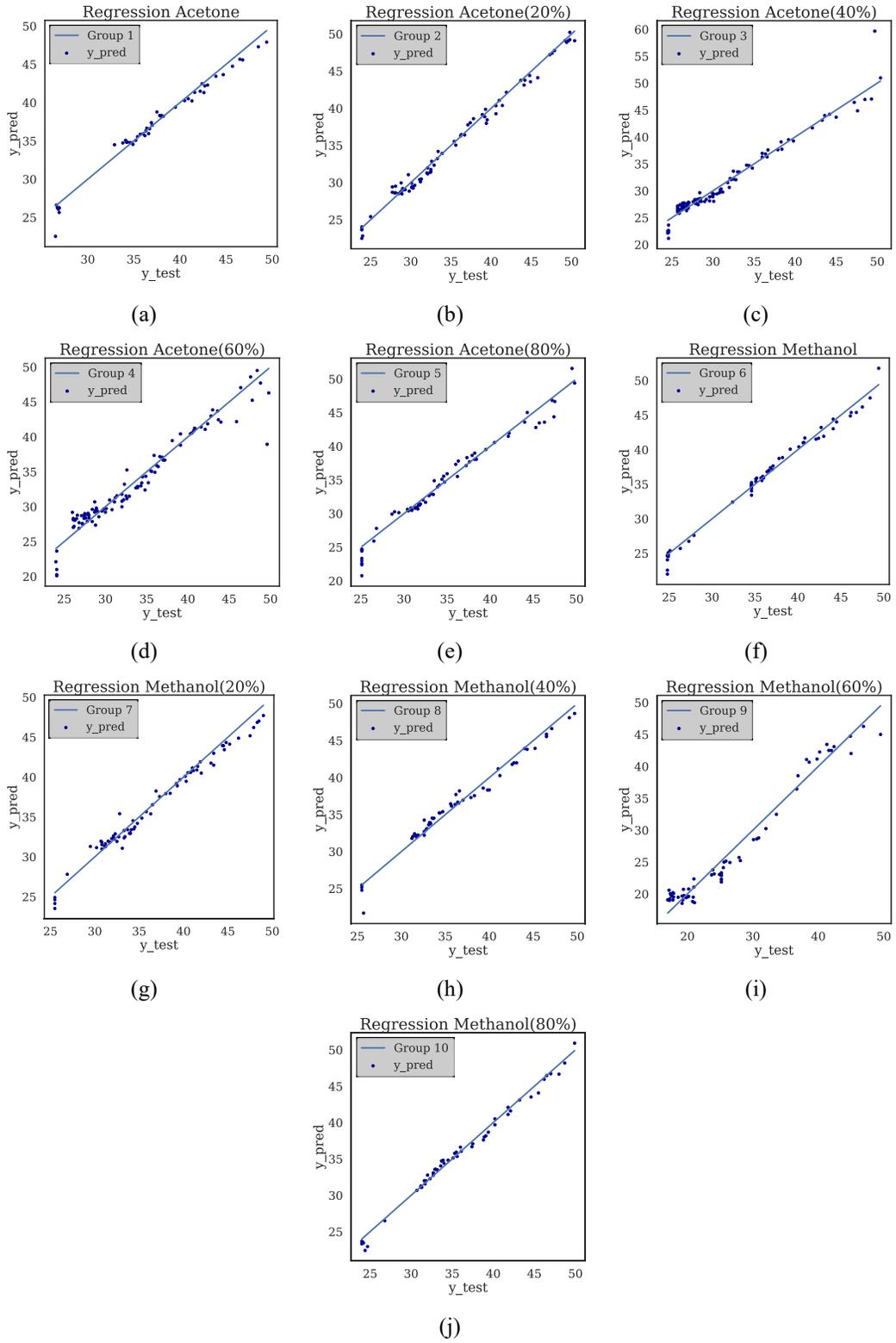


Figure 5.10 Linear regression result for all classes

5.3 Material Classification followed by Temperature Reporting

Two approaches are considered in this section:

One, to remove the effect of temperature from the sensor response and to classify the type of material regardless of the temperature it has. With that, temperature as an error is removed perfectly. In contrast to differential sensors which ignore the effect of temperature on materials, this method allows to consider its erroneous effect.

Reporting both material type and temperature of the material at the same time is the second approach that is the main focus of this study. The schematic diagram of the final algorithm is shown in Figure 5.11.

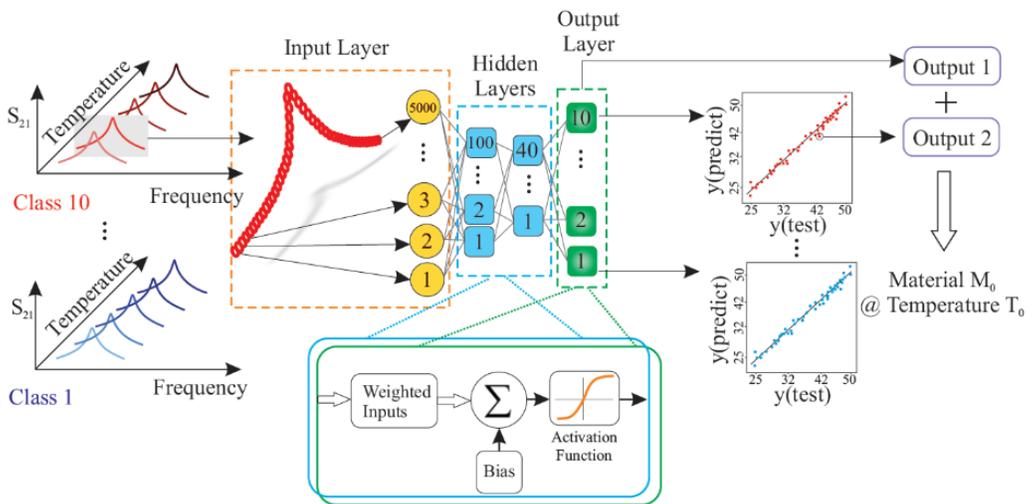


Figure 5.11 Machine learning flow from input data at varying temperature to reporting classified material with temperature

The S_{21} profiles are recorded while temperature changes as an environmental effect. A stack of S_{21} profiles, with respect to their labels, create the total dataset as explained in detail in Chapter 4 with a snapshot shown in Figure 5.12.

	s.1	s.2	s.3	s.4	s.5	s.6	s.7	s.8	s.9	...	s.4995	s.4996	s.4997	s.4998	s.4999	s.5000	classes	label	Temp
i.203	-36.190	-36.198	-36.210	-36.200	-36.194	-36.196	-36.170	-36.168	...	-38.770	-38.749	-38.798	-38.783	-38.787	-38.800		7	Methanol20%	44.4
i.630	-36.645	-36.636	-36.635	-36.634	-36.636	-36.619	-36.627	-36.613	...	-37.121	-37.129	-37.130	-37.126	-37.138	-37.143		1	Acetone	36.2
i.388	-36.402	-36.388	-36.386	-36.379	-36.375	-36.388	-36.362	-36.353	...	-38.703	-38.684	-38.704	-38.706	-38.713	-38.726		9	Methanol60%	19.7
i.426	-36.417	-36.429	-36.412	-36.418	-36.403	-36.408	-36.410	-36.394	...	-39.044	-39.029	-39.028	-39.041	-39.047	-39.043		3	Acetone40%	28.4
i.096	-36.096	-36.104	-36.096	-36.085	-36.078	-36.085	-36.068	-36.066	...	-39.178	-39.177	-39.166	-39.163	-39.176	-39.171		7	Methanol20%	30.3
i.470	-36.446	-36.443	-36.447	-36.453	-36.430	-36.433	-36.424	-36.418	...	-38.888	-38.912	-38.895	-38.884	-38.903	-38.905		3	Acetone40%	34.5
i.550	-36.517	-36.532	-36.539	-36.516	-36.516	-36.509	-36.506	-36.515	...	-38.596	-38.601	-38.590	-38.581	-38.590	-38.563		3	Acetone40%	45.0
i.705	-36.717	-36.698	-36.699	-36.713	-36.691	-36.681	-36.694	-36.666	...	-36.686	-36.691	-36.696	-36.718	-36.706	-36.721		1	Acetone	49.3
i.635	-36.620	-36.627	-36.634	-36.634	-36.616	-36.623	-36.613	-36.615	...	-38.546	-38.537	-38.559	-38.589	-38.551	-38.547		4	Acetone60%	38.3
i.605	-36.592	-36.592	-36.591	-36.594	-36.593	-36.583	-36.582	-36.574	...	-38.705	-38.727	-38.710	-38.723	-38.732	-38.727		4	Acetone60%	26.6
i.264	-36.243	-36.242	-36.239	-36.233	-36.251	-36.238	-36.217	-36.225	...	-38.898	-38.897	-38.915	-38.926	-38.920	-38.918		8	Methanol40%	32.4

Figure 5.12 Dataset snapshot from Python

Then these recorded values are fed into a deep neural network, including two hidden layers, which classifies the type of each recorded S_{21} based on its features. The relevant python code is also given in Code 5.5 below.

In order to report the measurement temperature, the following approach is devised to be combined with the classification. Since various materials (classes) react differently to temperature, it is much simpler to find a relationship between measurement temperature and profiles in each class rather than correlating the temperature to all recorded profiles. Therefore, a linear regression algorithm is trained based on each individual class, resulting in a total of 10 regression lines. Regression line of each class will be called to map input S_{21} to a specific temperature once the input S_{21} is found to belong to this class.

As a result, when the classification neural network and the linear regression algorithm are cascaded, the output of the classifier, as shown in Figure 5.11, determines which regression expression, that has already been trained, needs to be applied on the input S_{21} profile to help predict proper temperature value.

As an example, a few randomly selected input S_{21} profiles, which cover the whole classes, are tested with this cascaded scheme. The corresponding class of every individual input S_{21} as well as the predicted classes are shown in **Error! Reference source not found..** In addition, the corresponding predicted temperatures for each S_{21} are compared with the actual values as shown in Figure 5.14.

```

1. Xdata = dataset.iloc[:, 0:5000].values
2. ydata = dataset.iloc[:, 5003]
3. ydataclass = dataset.iloc[:, 5004]

4. print(Xdata.shape)
5. yclass_actual = []
6. yclass_pred = []

7. Temp_pred = []
8. Temp_actual = []
9. ind = np.arange(1,3400,10)
10. ind_arr = np.asarray(ind)
11. for i in range(ind_arr.shape[0]):
12.     index = ind_arr[i]
13.     X_rand = Xdata[index].reshape(1, -1)
14.     Temp_actual.append(ydata[index])
15.     X_input = sc.transform(X_rand)

16. y_pred_class = classifier.predict_classes(x=X_input)
17. # y_pred_class = classifier.predict_classes(X_test[1].reshape(1,-1))
18. y_list = y_pred_class.tolist()
19. i = y_list[0]
20. yclass_pred.append(y_list[0])
21. yclass_actual.append(ydataclass[index])

22. def sreg(i):
    switcher={
    0: regressor1.predict(X_rand),
    1: regressor2.predict(X_rand),
    2: regressor3.predict(X_rand),
    3: regressor4.predict(X_rand),
    4: regressor5.predict(X_rand),
    5: regressor6.predict(X_rand),
    6: regressor7.predict(X_rand),
    7: regressor8.predict(X_rand),
    8: regressor9.predict(X_rand),
    9: regressor10.predict(X_rand),
    }
    return switcher.get(i,"Invalid Input")

23. Temp_pred.append(sreg(i))

```

Code 5.5 Python code to feed output of classification algorithm into input of linear regression

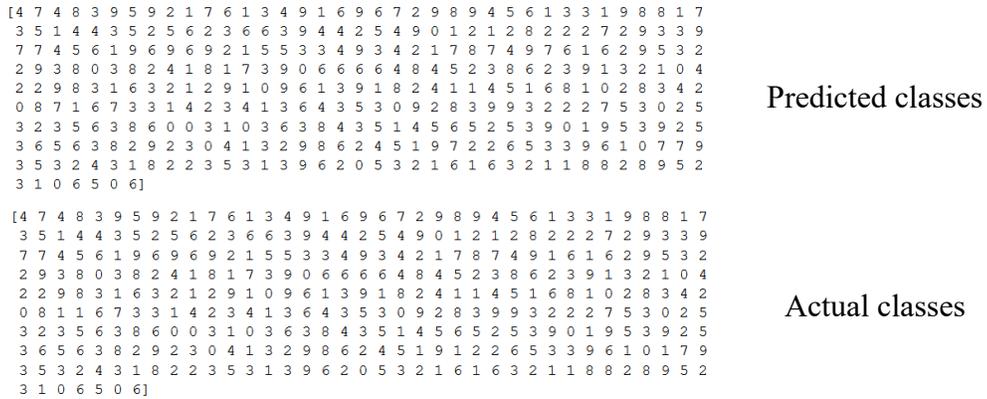


Figure 5.13 Predicted vs. Actual classes of 10 classes in use

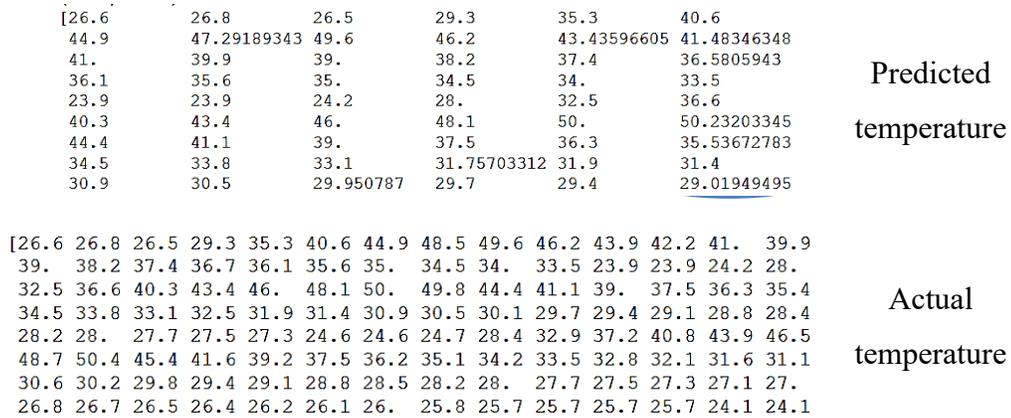


Figure 5.14 Temperature prediction vs. actual values for 10 classes in use

Combining all predictions made from the randomly selected S₂₁ profiles as input data points and assessing their accuracy is given in the following confusion matrix that elaborates on the relationship between the target classes of 10 material types and the predicted classes. It is noteworthy that the class of all input data is correctly recognized. Also, their predicted temperature is plotted in accordance with their target temperature in Figure 5.15 that exhibits a high R² correlation of 98%. This confirms that the proposed algorithm is able to discriminate the materials as

well as report the status of the environmental (here, temperature) with the use of cascaded Neural Network and Linear Regression.

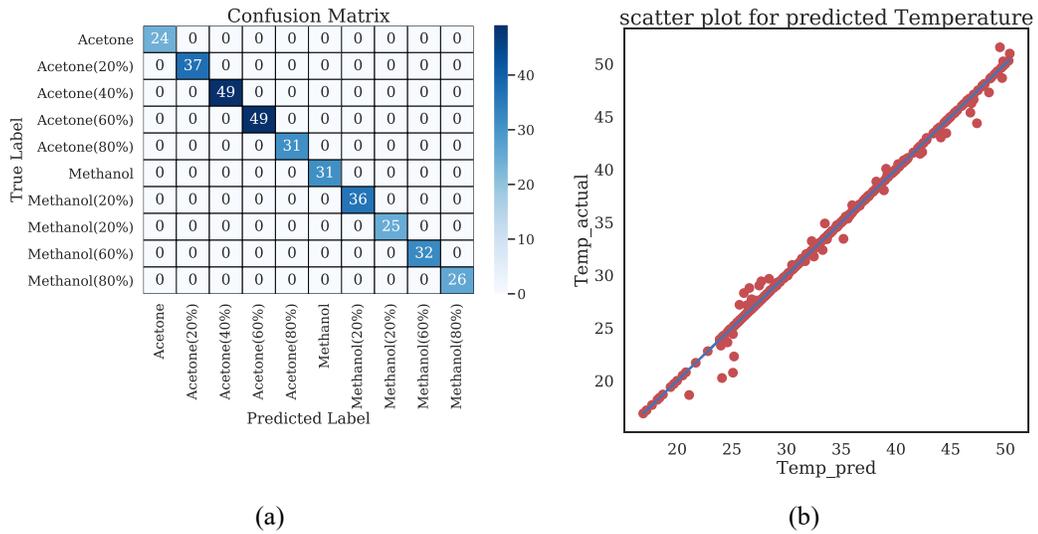


Figure 5.15 (a) Confusion matrix for classification of methanol/acetone in water samples, (b) Temperature reporting for randomly selected test materials within all classes

5.4 Conclusion

In this chapter, two important challenges in microwave sensor systems are addressed. At first, the environmental impacts are removed from the sensor response that enabled it to maintain correct interpretation of material regardless of the temperature of measurement. This is performed using various algorithms including decision tree, K-nearest neighbors, and neural network, each with specific advantages and disadvantages. All in all, neural network is chosen to work with, because it is versatile and provides higher accuracy rate compared with the other approaches. The outcome of the first step was the class of materials the measured materials belong to.

Chapter 5: ML Algorithms to Characterize Materials with Varying Temperature

In the next step, the temperature of measurement was reported. Therefore, a unique interpretation system was designed that used the output of the first stage as the input of the second stage, wherein a linear regression line is used for each class that correlates each temperature with every single measured datapoint. This relationship is found to be linear for all classes.

The combined system of classification (using neural network) and temperature reporting (using linear regression for dataset limited to each class) is uniquely suitable to infer the type of material, its concentration, and also the temperature of measurement, all using machine learning.

Chapter 6

Hyperparameter Optimization

This section describes improvements in the present system of machine learning with the aim of reducing the limit of detection in sensing using AI. It is always a point of discussion in all sensors to which degree the sensor is able to discriminate different materials or presence of any contaminant in a solution. With lower limit of detection, these sensors can be introduced as competitive replacement candidates for the prevailing costly and labor-intensive methods. In order to examine the level of discrimination of SRR sensor, an experiment similar to the one in chapter 5 is devised but with lower concentrations of methanol-in-water of 1-5 % volume fraction for methanol. It is intrcutive to note that the level of methanol is significantly small and that the majority of samples are made up of water. This makes the sensor response to reside around high dielectric constant levels, close to that of water.

In order to verify the effectiveness of machine learnign algorithms, decision tree is first chosen to classify measured data points into 6 classes of 0 % (pure water), 1%, 2%, 3%, 4%, and 5% methaonl-in-water. It has inconsistent mapping in many classes, as shown in Figure 6.1, that all in all results in poor output of 79% accuracy for the low concentrations.

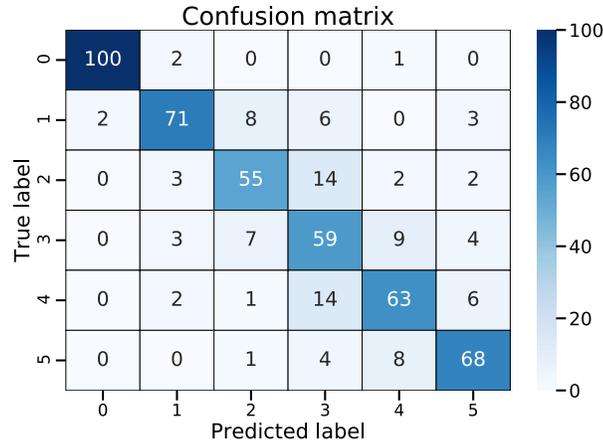


Figure 6.1 Confusion matrix for decision tree on low concentrations

In an attempt to verify the effectiveness of K-nearest neighbors, it has been found that mean error function becomes minimum at a specific K number as shown in Figure 6.2. The figure shows that the confusion matrix for this algorithm fails in proper assigning the measured samples since considerable number of inputs are mapped to Class 0 which is assumed to be pure water with 0% methanol in the solution. This means that the smaller the concentration, the harder it becomes for the decision tree to classify. All decisions are confused to be water samples, resulting in even lower overall accuracy rate of 66%.

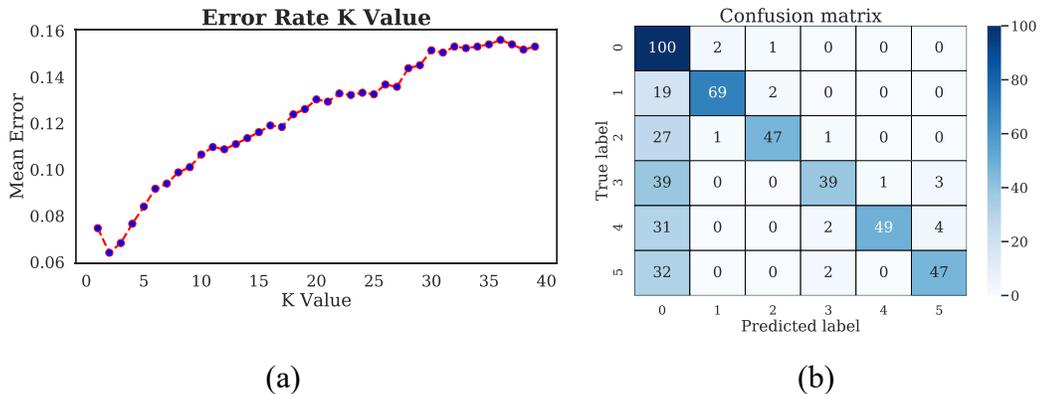


Figure 6.2 (a) Sweep on K number for optimum mean error, (b) Confusion matrix for K-nearest neighbors in classifying 0-5 % methanol-in-water samples for K = 2

This is the pivotal reason for involving multilayer perceptron (neural network, NN) to increase the accuracy level for lower concentration. The main reason is for this decision is the versatility of this NN embedded in its parameters. With this perspective, this chapter studies optimization on neural network in all its components.

6.1 Hyperparameter optimization

An essential part of deep learning is hyperparameter optimization, since neural networks have many parameters that need to be set. They include, but are not limited to:

- Number of batch-size and epochs
- Number of hidden layers
- Number of neurons in each layer
- Activation functions
- Loss functions

The performance, speed, and quality of the networks highly depend on these parameters. Therefore, it is necessary to optimize them. In order to optimize the network, an objective function should be considered. Validation accuracy, validation loss, f_1 score, etc. can be considered as an objective function that should be either maximized or minimized during the optimization process. This process yields a set of best parameters, which results in optimal model for the network.

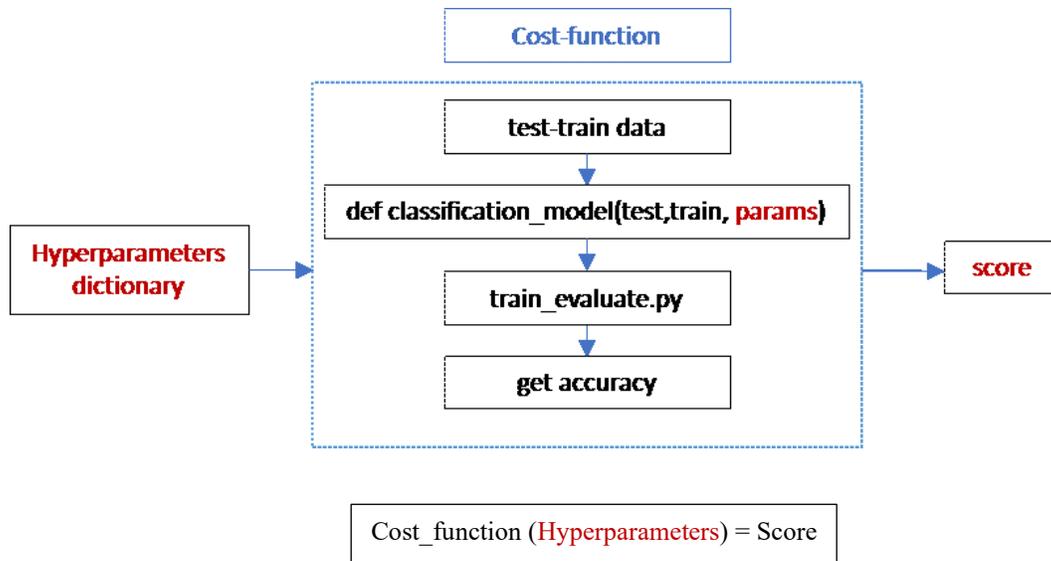


Figure 6.3 Schematic view of the hyperparametric study

In this study, models are built by Keras, and hyperparameter optimization is done using Talos as shown in Figure 6.3. To conduct the optimization, one must ensure that input data and parameters are passed into the argument of the defined function, see Code 6.1. A procedure similar to that used for building regular models is conducted except for the values in the model that will be chosen from the parameters' dictionary as given in Code 6.2.

Chapter 6: Hyperparameter Optimization

```
1. # first we have to make sure to input data and params into the function
2. def classification_model(x, y, x_val, y_val, params):

3.     model = Sequential()
4.     model.add(Dense(units = params['first_neuron'], input_dim=X_train.shape[1],
        activation=params['activation'],
        kernel_initializer=params['kernel_initializer']))
5.     model.add(Dropout(params['dropout']))
6.     model.add(Dense(units = params['second_neuron'],
        activation=params['activation'],
        kernel_initializer=params['kernel_initializer']))
7.     model.add(Dense(6, activation=params['last_activation'],
        kernel_initializer=params['kernel_initializer']))
8.     model.compile(loss=params['losses'],
        optimizer=params['optimizer'],
        metrics=['acc', talos.utils.metrics.f1_score])
9.     history = model.fit(x, y,
        validation_data=[x_val, y_val],
        batch_size=params['batch_size'],
        callbacks=[talos.utils.live()],
        epochs=params['epochs'],
        verbose=0)

10. return history, model
```

Code 6.1 Python code for hyperparametric study in MLP

After the keras model is built, initial parameters boundaries are set. Once the dictionary is fed to the model, a combination of parameters is picked from the dictionary in a single permutation and then excluded from next permutation, see Code 6.2.

Chapter 6: Hyperparameter Optimization

```
1. # then we can go ahead and set the parameter space
2. p = {'first_neuron':[75,100,300,500,600],
3.      'second_neuron':[20,30,40,50],
4.      'hidden_layers':[0,1, 2],
5.      'batch_size': [4,8,16,32, 128, 256],
6.      'epochs': [100,75,50,25,10],
7.      'dropout': [0],
8.      'kernel_initializer': ['uniform','normal'],
9.      'optimizer': ['Nadam', 'Adam','SGD','RMSprop','Adadelta'],
10.     'losses':['kullback_leibler_divergence','categorical_crossentropy',
11.              'binary_crossentropy','mean_squared_error'],
12.     'activation':['relu', 'elu'],
13.     'last_activation': ['softmax','sigmoid']}
```

Code 6.2 Dictionary of MLP parameters

Without adding any complexity to the code and learning new syntax, hyperparameter optimization can be done using Talos as shown in Code 6.3

```
talos.Scan(x, y, model, params).predict(x_test, y_test)
```

```
1. # and run the experiment
2. t = talos.Scan(x=x, y=y,
3.               model=classification_model,
4.               params=p, experiment_name='classification_model')
```

Code 6.3 Hyperparameter optimization scan using Talos

There are over 280,000 permutations inside the dictionary. After all permutations are scanned by Talos, it is time to analyze the results to decide on how to confine and change the parameters in the model. To have a better visibility of the categorical dataset returned by Talos another tool is used, called seaborn.

6.2 The Number of Neurons in the Hidden Layer

The number of hidden layers and the number of neurons in each layer is an important parameter at least to have an outlook of the network. In this study, the number of neurons for three cases is examined, when there is zero, one, and two hidden layers. First of all, to understand the topology of the network better, the number of layers needs to be determined. One can consider both the validation accuracy and validation loss as criteria to evaluate the performance of the network during the optimization process. Two hidden layers can be a reasonable choice to begin with, since it leads to a lower validation loss as shown in Figure 6.4.

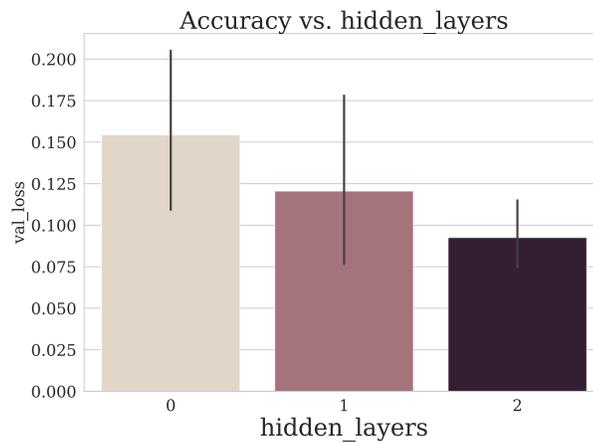


Figure 6.4 Hyperparameter optimization study on the hidden layer

Afterwards, the number of neurons in each layer should be tuned. The boundaries for the number of neurons in the first and second layers are set to [75, 100, 300, 500, 600] and [20, 30, 40, 50], respectively.

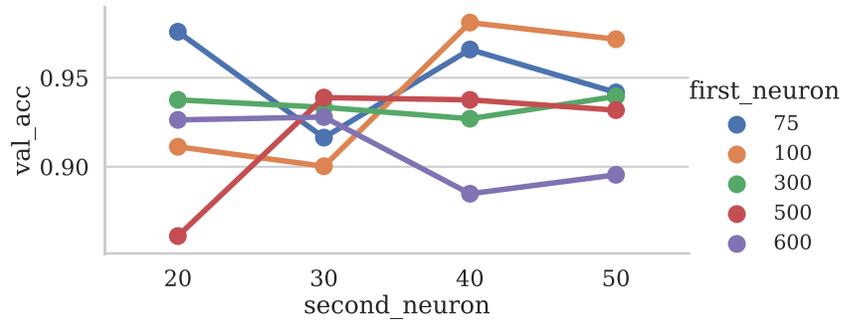


Figure 6.5 Hyperparameter optimization study with validation accuracy on neurons in the first and second layer

According to Figure 6.5, the maximum accuracy is achieved for two sets of neurons combinations. The first point of the blue curve [75, 20], and the third point of the orange curve [100, 40] both represent the best combination of neurons in each layer. One can choose either the first or the second combination. As shown in the Figure 6.6, the minimum value of the loss belongs to the same combination too.

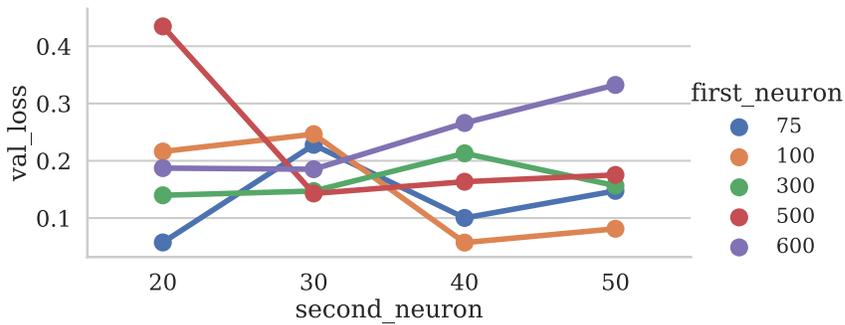


Figure 6.6 Hyperparameter optimization study on with validation loss on neurons in the first and second layer

6.3 Tuning Batch Size and Number of Epochs

The batch size is the number of data points that is shown to the network before any updates are applied to the weights. During training the network, this number is

one of the parameters that should be optimized. The number of times that the entire training dataset is introduced to the network is called epoch.

Here, the boundaries for the batch size and number of epochs are set to [4, 8, 16, 32, 128, 256] and [10, 25, 50, 75, 100], respectively, as shown in Figure 6.7. It is obvious that for the smaller batch sizes, the accuracy score is significantly higher than for the large batches. Also in terms of validation accuracy, from 8 to 32, the means are almost at the same level, see Figure 6.7 (b). At this step, the number of batch size is set to 8 or 16 because they provide good network performance.

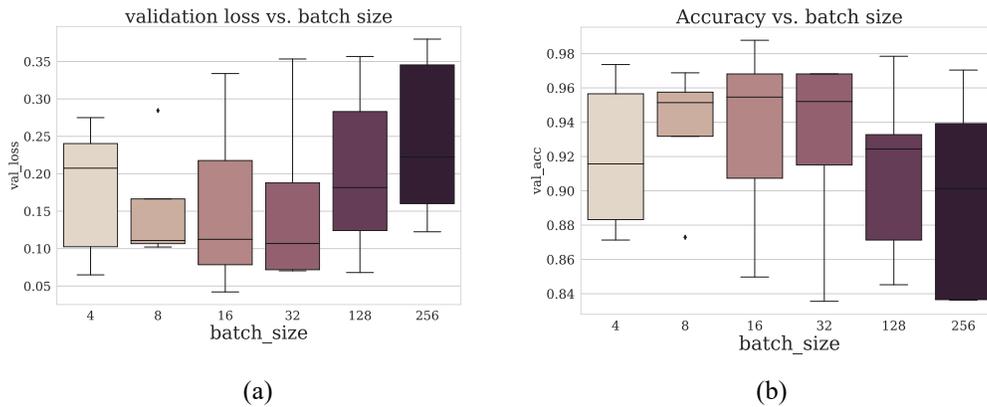


Figure 6.7 (a) Hyperparameter optimization study on the batch size with (a) validation loss and (b) accuracy loss

The following analysis is performed to set the best number of epochs after the tuning process, as shown in Figure 6.8. The higher values for epoch count, the better validation accuracy and performance of the network. On the other hand, more epochs require more resources in terms of computation time and hardware for training the network. Since the reasonable values for accuracy and loss are achieved for epoch = 100, there is no need to tune the epochs' parameter to higher values.

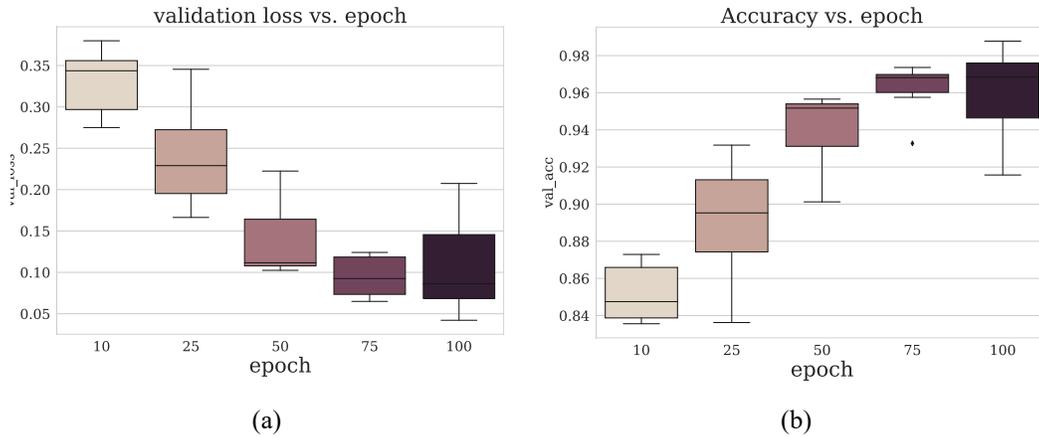


Figure 6.8 Hyperparameter optimization study on the number of epochs with (a) validation loss and (b) accuracy

6.4 Tuning the Kernel Initializer

The neural networks need to start with a set of weights that are updated in each iteration to better values. Usually, the argument used for passing initializers to layers is `Kernel_initializer`. There are different types of statistical distributions or functions, such as `RandomUniform`, `RandomNormal`, `TruncatedNormal`, and other, that are used for weights initialization. Here, two types of statistical distributions are employed to generate numbers to be used as the starting weights. Neural network layers, as non-linear maps, transfer the data from a given input space to another space using kernel initializer. As shown in Figure 6.8, two parameters [uniform, normal] are considered to evaluate which `Kernel_initializer` works better.

Since both initializers behave similarly, it is hard to say which one has better performance from the validation accuracy on the y-axis of Figure 6.9. To gain more insight, let us take a look at the validation loss. With each parameter permutation, we are seeking smaller values for validation loss on the y-axis, see Figure 6.10.

Chapter 6: Hyperparameter Optimization

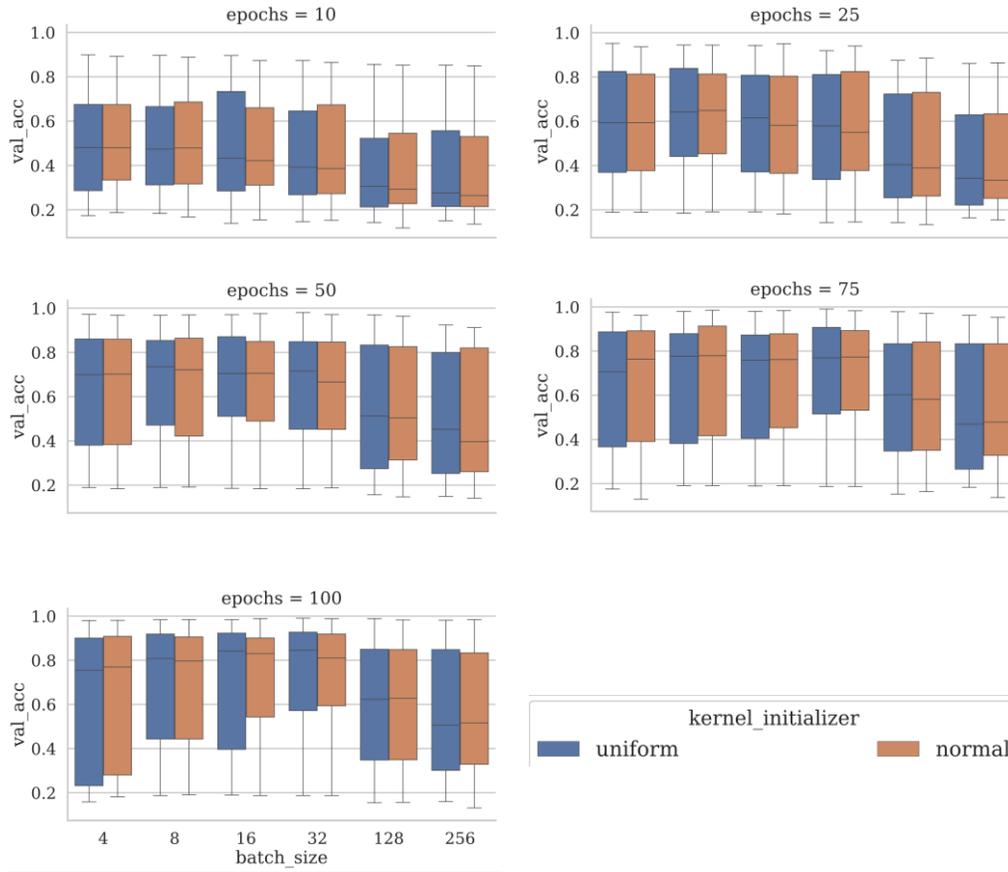


Figure 6.9 Hyperparameter optimization study on Kernel initializer with validation accuracy

Both uniform and normal kernel initializers are doing a great job in keeping the validation loss down throughout all epochs and more or less have the same behavior for different batch sizes. The validation loss is minimized if the number of epochs is set to 100, and the number of batch size equals 8, with either a uniform or normal as kernel initializers. There is an option to keep both of them to the final stage to find the best permutation.

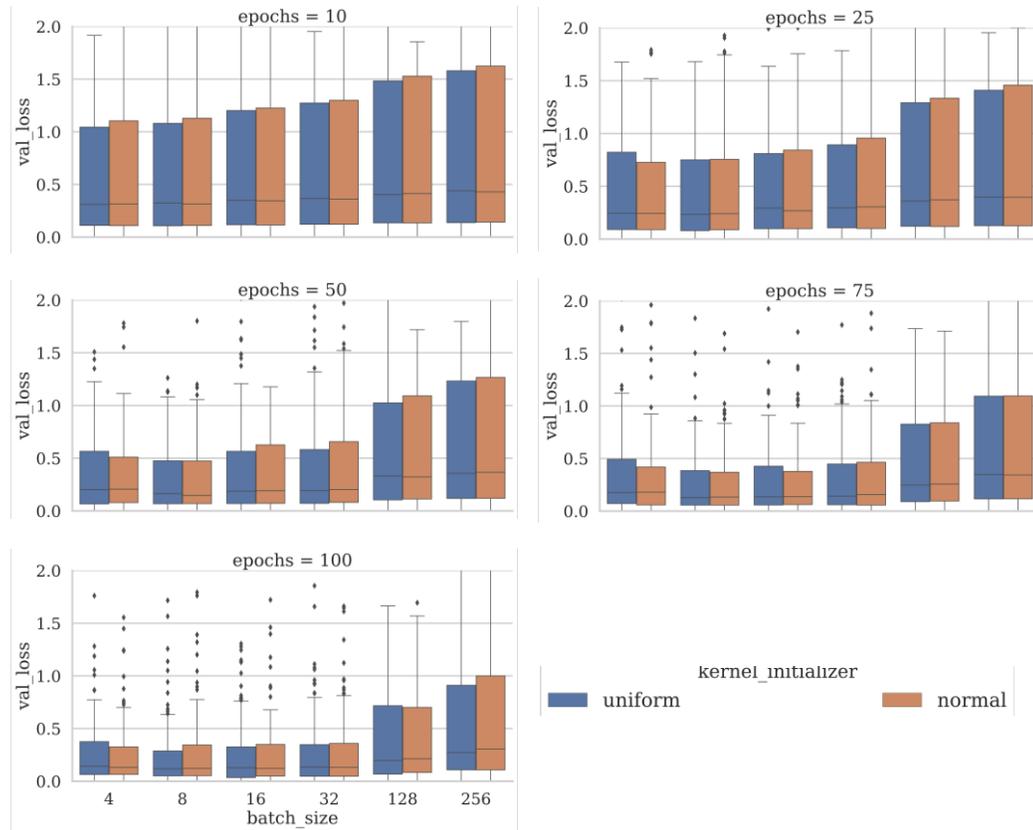


Figure 6.10 Hyperparameter optimization study on Kernel initializer with validation loss

6.5 Tuning the Training Optimization Algorithm

Optimization is a core component of machine learning algorithms. In other words, most machine learning algorithms build an optimization model to find an extremum of an objective function. Optimizers update the weight parameters to minimize the loss function. This function provides feedback for the optimizer, whether it is moving in the right direction to reach the global minimum or not. The optimization methods are split into three main categories:

- First-order optimization method is most common, with stochastic gradient descent (SGD) and its variants used most often.
- High-order optimization methods, such as Newton's method.

- Heuristic derivative-free optimization methods, e.g. methods that coordinate descent-based approaches.

For neural networks, and many other machine learning algorithms, gradient descent is the best choice to optimize the network. Although Keras offers various optimization algorithms, but they often come as a black box with no direct access to the optimizers' parameters. If one needs to conduct a parametric sweep related to a specific optimizer, the optimizer should be defined as a function through the network. In this thesis, the network is optimized with different optimization algorithms, each with default parameters defined by Keras. However, a brief study is presented in this section to provide a better illustration of how optimizers work.

In this regard, the following parameters are defined. To minimize an objective function $J(\theta)$, the model's parameters $\theta \in \mathbb{R}^d$ will be updated in the opposite direction of $\nabla_{\theta}J(\theta)$. The learning rate η determines the size of the step of descent.

6.5.1 Stochastic gradient descent

Gradient descent approaches differ in the amount of data that the optimizer uses to calculate the gradient of the objective function [86]–[88]. These methods include batch gradient descent, mini-batch gradient descent, and stochastic gradient descent. Based on how much data the optimizer uses, there is a trade-off between the time and the accuracy to update the parameters. Although batch gradient descent guarantees convergence to the global minimum for a convex error-surface and a local minimum for a non-convex surface, it is too slow; since it needs to calculate gradients for the whole dataset to updates just once. Besides, the mini-batch size does not guarantee a good convergence. In contrast with batch gradient descent, stochastic gradient descent conducts a parameter update for each training example x^i and label y^i .

$$\theta = \theta - \eta \cdot \nabla_{\theta}J(\theta; x^i; y^i) \tag{6.1}$$

Thus, it is faster than batch gradient descent. Similar to batch gradient descent, there is no guarantee to reach the global minimum since it is difficult to choose an appropriate learning rate using the same value for all parameters. In the following, four algorithms that are used to address the aforementioned problem are introduced.

6.5.2 Adadelta

This optimization algorithm is similar to AdaGrad in its principles. The gradient descent is replaced with more concise notation of $\mathbf{g}_{t,i}$ as follows:

$$\mathbf{g}_{t,i} = \nabla_{\theta} J(\theta_{t,i}) \quad 6.2$$

where $\mathbf{g}_{t,i}$ is the partial derivative of the objective function at time step t .

The parameter update for $\theta_{t,i}$ is as follows, where the learning rate η becomes modified as given in 6.4:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot \mathbf{g}_{t,i} \quad 6.3$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot \mathbf{g}_{t,i}, \quad 6.4$$

where $G_{t,ii} \in \mathbb{R}^{d \times d}$ is a diagonal matrix whose elements are sum of the squares of the gradient, and ϵ is the smoothing parameter to avoid zero denominator. It is noteworthy that, in AdaGrad, the accumulated gradient ($G_{t,ii}$) becomes larger and larger in the denominator as the training time increases (see 6.4). This results in a zero learning-rate, which makes further updates meaningless. However, Adadelta algorithm restricts the accumulation window of past gradients to a fixed value. Therefore, to prevent the accumulation sum from constantly growing during training, $G_{t,ii}$ is replaced with the decaying average over past squared gradients $E[\mathbf{g}^2]_t$ as follows:

$$\Delta \theta_t = - \frac{\eta}{\sqrt{E[\mathbf{g}^2]_t + \epsilon}} \cdot \mathbf{g}_{t,i} \quad 6.5$$

The denominator of the above equation is the root mean square (RMS) error of the gradient descent as follows:

$$\sqrt{E[g^2]_t + \epsilon} = RMS[g]_t \quad 6.6$$

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} \cdot g_t \quad 6.7$$

Similar to equation 6.6 we can define:

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon} \quad 6.8$$

Since the learning rate is an unknown quantity, it is replaced with the RMS of parameter updates until the previous time step ($RMS[\Delta\theta]_{t-1}$) as follows:

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} \cdot g_t \quad 6.9$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \quad 6.10$$

According to equation 6.9, Adadelta does not need to set a default learning rate η . Also, it changes the trajectory of gradient accumulation towards an exponentially decaying average to avoid near extreme-zero η .

6.5.3 RMSProp Gradient Descent

Similar to Adadelta, RMSProp [89]–[91] tries to solve the same problem where AdaGrad suffers. Both Adadelta and RMSProp are suitable for non-stationary and non-convex problems. However, both methods have a poor performance in the late training stage since the update process may be repeated around the local minimum.

Here, in order to evade the drastically diminishing learning rate, $G_{t,ii}$ is now replaced with another expression as follows:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_{t,i} \quad 6.11$$

$$E[g^2]_t = (1 - \gamma)g_{t-1}^2 + \gamma \cdot g_t \quad 6.12$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1-\gamma)g_{t-1}^2 + \gamma \cdot g_t + \epsilon}} \cdot g_{t,i}, \quad 6.13$$

where γ is suggested to be set to 0.9, while a reasonable default value for the learning rate is 0.001. In RMSProp, learning rate will be different for each parameter since it is adjusted automatically during the training time.

6.5.4 Adam— Adaptive Moment Estimation

Adam optimizer is one of the most popular gradient descent optimization algorithms [92]. It combines the adaptive learning rate and momentum methods simultaneously. Similar to Adadelta and RMSProp, Adam stores an exponentially decaying average of the past squared gradient (v_t), and also similar to the momentum methods, keeps an exponentially decaying average of past gradients (m_t) at the same time:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad 6.14$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad 6.15$$

where m_t and v_t are the first- and second-moment of the gradients, respectively. The default values for hyperparameters β_1 , β_2 are suggested to be set to 0.9 and 0.999, respectively. There is a potential problem with m_t and v_t updated values as they are biased towards zero with initial zero vector. To resolve this problem, bias-corrected estimates are proposed as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad 6.16$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad 6.17$$

$$\theta_{t+1} = \theta_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad 6.18$$

where ϵ is suggested to be 10^{-8} .

6.5.5 Nadam - Nesterov-accelerated Adaptive Moment Estimation

Similar to Adam, which is a combination of adaptive learning rate and momentum methods, Nadam is the combination of adam and Nesterov-accelerated gradients (NAG) [93]–[96]. Again, please recall that momentum methods keep an exponentially decaying average of past gradients (m_t), and Adam stores an exponentially decaying average of the past squared gradient (v_t). In order to incorporate NAG into Adam, we need to modify its momentum term m_t .

First, let us consider the expansion of the Adam update rule with inserting 6.14 in 6.16, and plugging it in 6.18 as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \left(\frac{\beta_1 m_{t-1}}{1 - \beta_1^t} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \quad 6.19$$

Please note that the problem with bias, explained in Adam, can be corrected within equation 6.19 if $\frac{m_{t-1}}{1 - \beta_1^t}$ can be somehow replaced with $\hat{m}_{t-1} = \frac{m_{t-1}}{1 - \beta_1^{t-1}}$ assuming that $\beta_1^{t-1} = \beta_1^t$, which holds true in many practical cases. Now, we can add Nesterov momentum by replacing the bias-corrected estimate of the momentum vector of the previous time step (\hat{m}_{t-1}) with the bias-corrected estimate of the current momentum vector (\hat{m}_t), which results in Nadam update rule as follows [97]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \left(\beta_1 \hat{m}_{t-1} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \quad 6.20$$

6.5.6 Comparison of Optimizers

In this section, we present a more in-depth study of various optimizers in the neural network, including Nadam, SGD, Adadelata, Adam, and RMSProp. As shown in Figure 6.11, based on the discussion presented earlier around the performance of the above-mentioned optimizers, SGD in all combinations of epochs and batch sizes has a poor performance. The next four optimizers compete with each other, but Adam stands out from the rest.

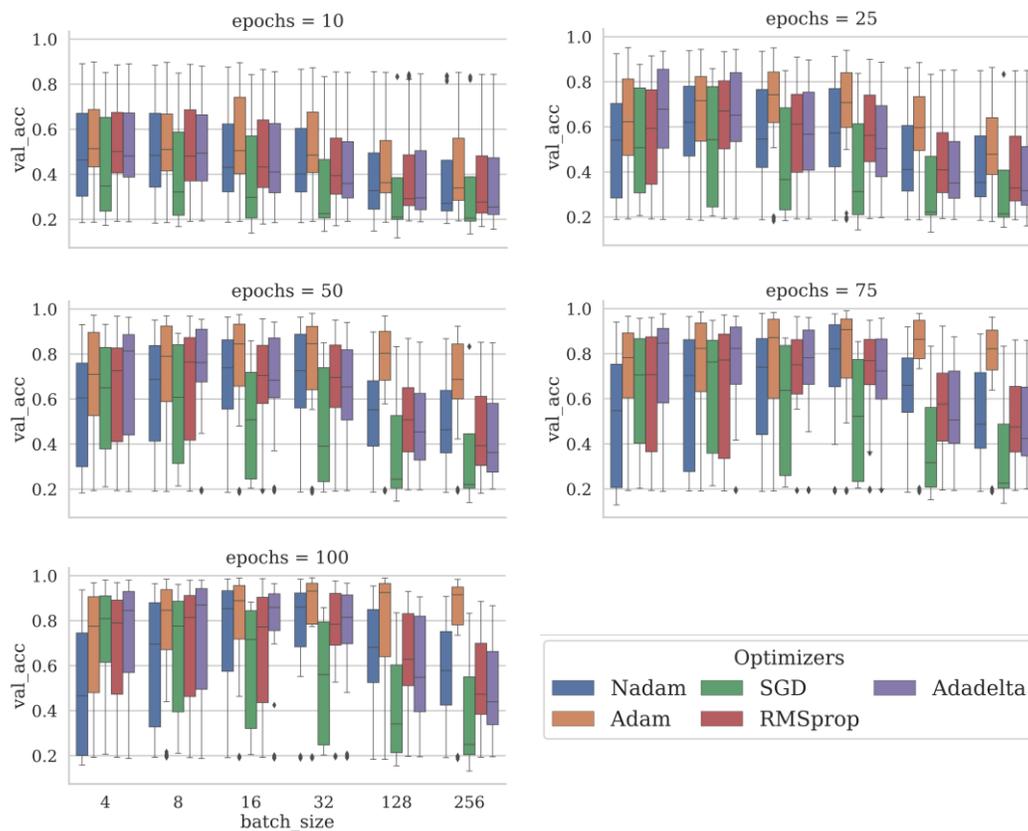


Figure 6.11 Hyperparameter optimization study on various optimizers, including Nadam, SGD, Adadelata, Adam, and RMSprop using validation accuracy.

From the previous results of parametric sweep epoch = 100 and batch size = 8 and 16 are of interest; therefore let us narrow the optimization processes to the last subplot in Figure 6.12. Adam and Nadam outperformed the others in batch size ranging from [4, 8, and 16]. To be more confident about the accuracy of our

optimization, another set of permutations for both epoch and these two optimizers has been performed.

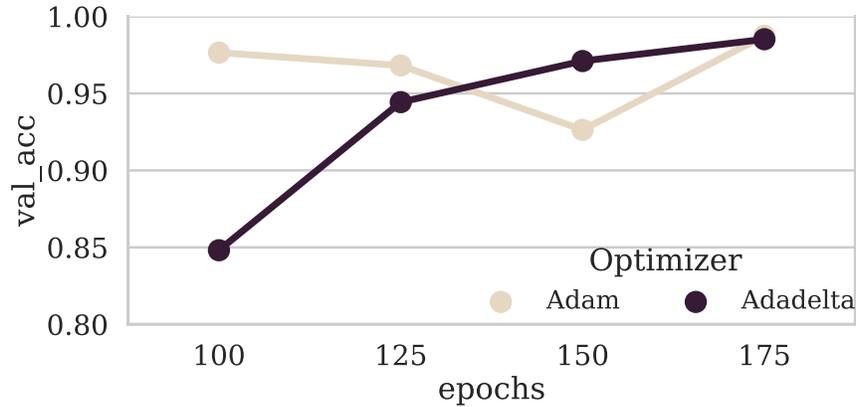


Figure 6.12 In-depth comparison of best optimizers, Adam vs. Adadelta, using validation accuracy in different number of epochs

Figure 6.12 shows for epoch equals to 100, Adam works well; however, with the increase in the number of epochs, Adadelta gets closer to Adam. The trade-off between time and accuracy makes us decide which one is fit for our network. It is noteworthy to mention that the value of 0.98 is reasonably acceptable as a validation accuracy; thus, it does not worth going for a larger epoch number in Adadelta since it requires a significantly longer time to train the network.

6.6 Choice of Loss Function for Training of Deep Neural Networks

As discussed in the previous section, deep learning neural networks are trained with the aid of stochastic gradient descent optimization algorithms. A metric is required to ensure that we are on the right track in the optimization process. During the process, the error for the current state of the model must be calculated continuously. Therefore, an error function, called "loss function," must be defined to estimate the loss of the model. Through the process, weights are updated to reduce the loss on the next evaluation [98]–[101].

In general, neural network models learn to find the best mapping model from input data to output using examples, and the choice of loss function needs to be compliant with the type of prediction model. Supervised neural networks are divided into two main categories: classification and regression. In this section, we are dealing with multi-class classification model; as a result, an appropriate loss function is the one that is chosen based on the shape of the output layer and classification model.

In this section, four loss functions are considered to decide which one is the best fit for the model. They are: Mean Squared Error (MSE), binary crossentropy, categorical crossentropy, and Kullback-Leibler divergence. In the following subsections $L(y, \hat{y})$ represents the loss function, where y is the actual value of the output data and \hat{y} is the value predicted by the model.

6.6.1 Mean Squared Error Loss

Mean square error loss is widely used for regression problems and is equal to the average of the squared differences between the predicted and actual values:

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^N (y - \hat{y}_i)^2 \quad 6.21$$

According to the previous equation, $(y - \hat{y}_i)^2$ shows that larger errors are more significant than smaller ones. The mean squared error loss function can be used in Keras by specifying `'mse'` or `'mean_squared_error'` as the loss function when compiling the model.

6.6.2 Binary Cross-Entropy Loss

Binary Cross-Entropy Loss is commonly used for classification problems that the input is categorized with only two labels. This loss function will remain the same unless it becomes updated by a better (lower) value. In predicting class 1, a score is calculated by cross-entropy that includes the average difference between

the actual and predicted probability distributions. The final loss is obtained by averaging these class-wise errors.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^N (y * \log(\hat{y}_i) + (1 - y) * \log(1 - \hat{y}_i)) \quad 6.22$$

Cross-entropy can be specified as the loss function in Keras by specifying *'binary_crossentropy'* when compiling the model.

6.6.3 Multi-Class Cross-Entropy Loss

Cross entropy is the default loss function to use for multi-class classification problems.

Here, score is calculated by cross-entropy that includes the average difference between the actual and predicted probability distributions for all classes. The score is minimized and a perfect cross-entropy value is 0. The final loss is obtained by averaging these class-wise errors.

$$L(y, \hat{y}) = - \sum_{i=0}^M \sum_{j=0}^N (y_{ij} * \log(\hat{y}_{ij})) \quad 6.23$$

Cross-entropy can be specified as the loss function in Keras by specifying *'categorical_crossentropy'* when compiling the model.

6.6.4 Kullback-Leibler Divergence Loss

Kullback-Leibler divergence, or KL divergence for short, measures how much one probability distribution differs from another. Practically, KL divergence is similar to cross-entropy. In the case of approximating a more complex function rather than multi-class classification, the KL divergence loss function is widely used. For instance, it is used in an auto-encoder to learn a model to reconstruct an original input. However, it can be used in multi-class classification problems with equal functionality compared with multi-class entropy.

$$D_{KL}(y||\hat{y}) = \sum y * \log(y/\hat{y}) \quad 6.24$$

KL divergence loss can be used in Keras by specifying *'kullback_leibler_divergence'* in the *compile()* function.

6.6.5 Comparison of Loss Functions

To find the best loss function for the model, four different functions are chosen in the parameter dictionary. With a glance at the plots in Figure 6.13, it is clear that binary cross-entropy keeps the validation accuracy high throughout all epochs and batch sizes. The best choice in all steps of optimization strongly depends on the type of data and the model one works with. Having basic knowledge about all the parameters that are set to the parameters' dictionary helps to select their best combination. However, in the end, it is the outcome of hyperparameter optimization that sets the optimum parameters for the best network performance.

Chapter 6: Hyperparameter Optimization

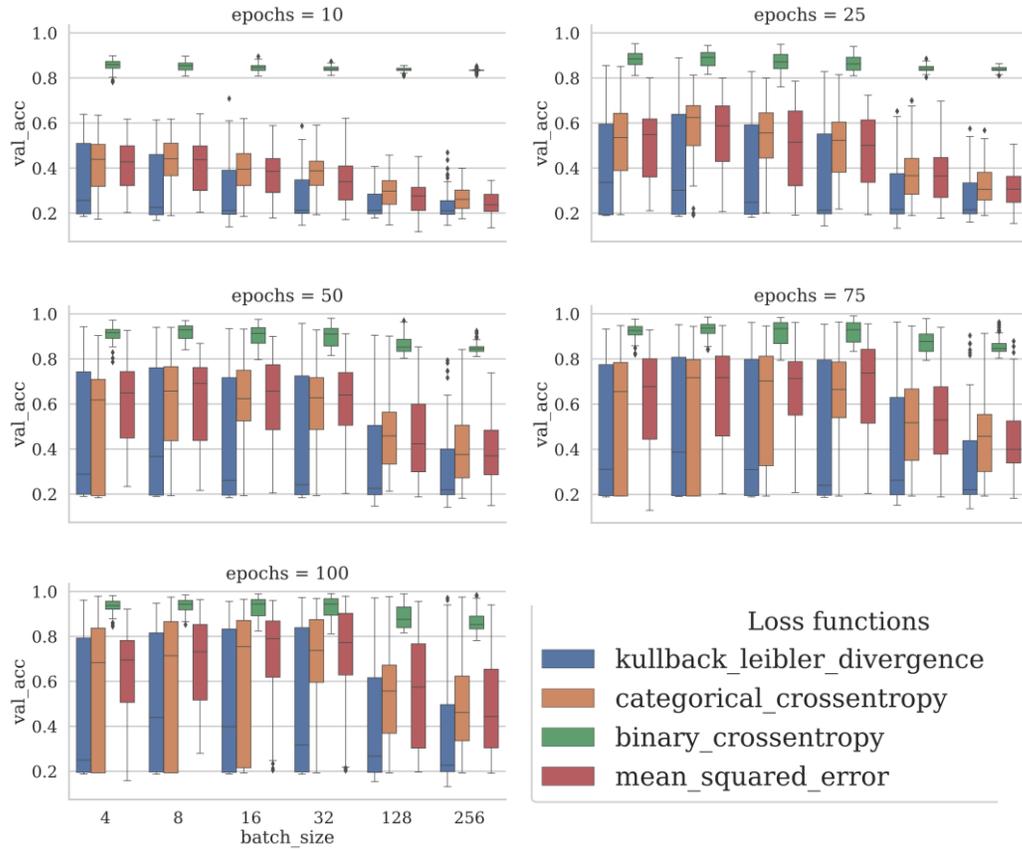


Figure 6.13 Hyperparameter optimization study on various loss functions using validation accuracy

6.7 Tuning of the Neural Activation Function

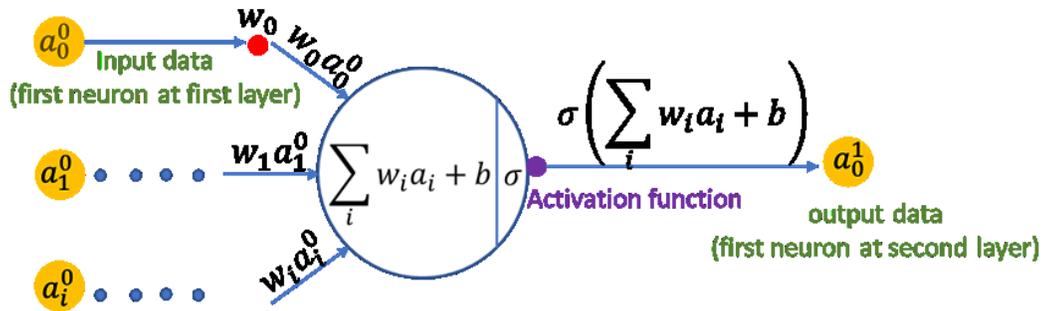


Figure 6.14 Schematic of the activation function

Neural networks are based on algorithms designed to recognize patterns in complex data. In simpler language, neurons in the network are interconnected. To represent them in math language, we simply assign a number to each neuron, and each connection holds a weight. Activation functions are mathematical equations that are applied on the neurons in the network to determine whether they should be activated or not [102]–[106]. They help the network learn intricate patterns in the data. Besides, they add nonlinearity to the multiple layers of the network without which hidden layers become meaningless, and any layered network is equivalent to the network with just one single layer. Throughout the network, each neuron has some activation value ranging from 0 to 1. Zero here means that specific neuron is not activated ("fired") since it is not relevant for the model prediction.

Let's revisit the activation function analytically. From here on, each neuron has an activation function σ and if it is connected to a new neuron, it has a weight w as follows:

$$w_1 a_1 + w_2 a_2 + \dots = \sum_1^n w_i a_i \quad 6.25$$

This means that n inputs are weighted and added together to contribute to the neuron in the next layer. Next, we consider the bias b in the activation function. Similar to any other linear function that use a constant parameter to have a better fit, bias adjusts the neuron along with the weighted sum of the inputs. In the absence of bias, the model has a poor fit, as all lines without bias have to pass through the origin (0,0):

$$y = mx + c \rightarrow \text{biased weighted sum} = \sum(W_i \times a_i) + b \quad 6.26$$

The final equation subjects this neural internal activity to the activation function:

$$\sigma(w_1 a_1 + w_2 a_2 + \dots + b) = \text{new neuron} \quad 6.27$$

There is a comprehensive formula to produce all neurons in the subsequent layers:

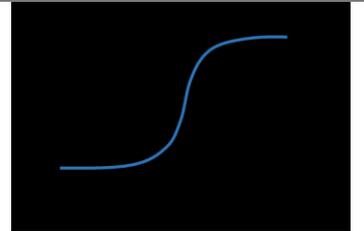
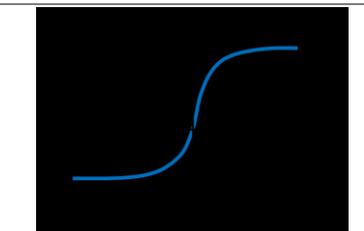
$$\sigma \left(\begin{bmatrix} w'_{0,0} & w'_{0,1} & \dots & & \begin{bmatrix} a_0^0 \end{bmatrix} & \begin{bmatrix} b_0 \end{bmatrix} \\ w'_{1,0} & w'_{1,1} & \dots & & \begin{bmatrix} a_1^0 \end{bmatrix} & \begin{bmatrix} b_1 \end{bmatrix} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ w'_{j,0} & w'_{j,1} & \dots & & \begin{bmatrix} a_n^0 \end{bmatrix} & \begin{bmatrix} b_n \end{bmatrix} \end{bmatrix} \right) \quad 6.28$$

where $a_{neuron}^{(layer)}$ represents the location of specific activation in the network. The subscript shows the location of neuron in each layer, while the superscript shows indicates the layer this neuron belongs to. For example, $a_2^{(1)}$ corresponds to the third neuron in the second layer (start from 0) of the network. Each weight in the first matrix is shown in the format of $w_{to, from}^{layer}$. For example, $w_{j=2, k=3}^{l=2}$ means *from* neuron $k+1=4$ in the previous layer $l=2$ (second layer) *to* neuron $j+1=3$ in the third (k) layer. To describe all neurons in all layers, the following compact formula is used:

$$a^{(l)} = \sigma(Wa^{(l-1)} + b) \quad 6.29$$

which says that all activations a from layer $l - 1$ are multiplied by all weights W (connecting each neuron from the previous layer to the next one), and a bias value (b) is added. This summation is passed as an argument to the activation function to provide all the activations for the next layer l . Here a summarized table is provided to show the features of these four activation functions that are considered in the dictionary.

Table 6.1 Activation function definitions

<p>ReLU Function</p> $\sigma(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
<p>ELU Function</p> $\sigma(x) = \begin{cases} x & \text{for } x > 0 \\ \alpha(e^x - 1) & \text{for } x < 0 \end{cases}$	
<p>Sigmoid Function</p> $\sigma = \frac{1}{1 + e^{-x}}$	
<p>Softmax Function</p> $\sigma(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^j \exp(x_j)}$ <p>for $i = 0, 1, 2, \dots, j$</p>	

Once the features of activation functions (σ) are known, an individual study is performed on the four different types of functions that were used in the optimization process. It should be noted that in the hidden layers, it is better to use activation functions such as Relu since they are not only computationally efficient but also avoid vanishing gradient descent problem and allow the network to converge faster. Here both Relu and elu provide high validation accuracy; thus, there is no preference. Elu is fitted to the network since, at epoch 100 and batch size ranging from 4 to 32, it performs slightly better.

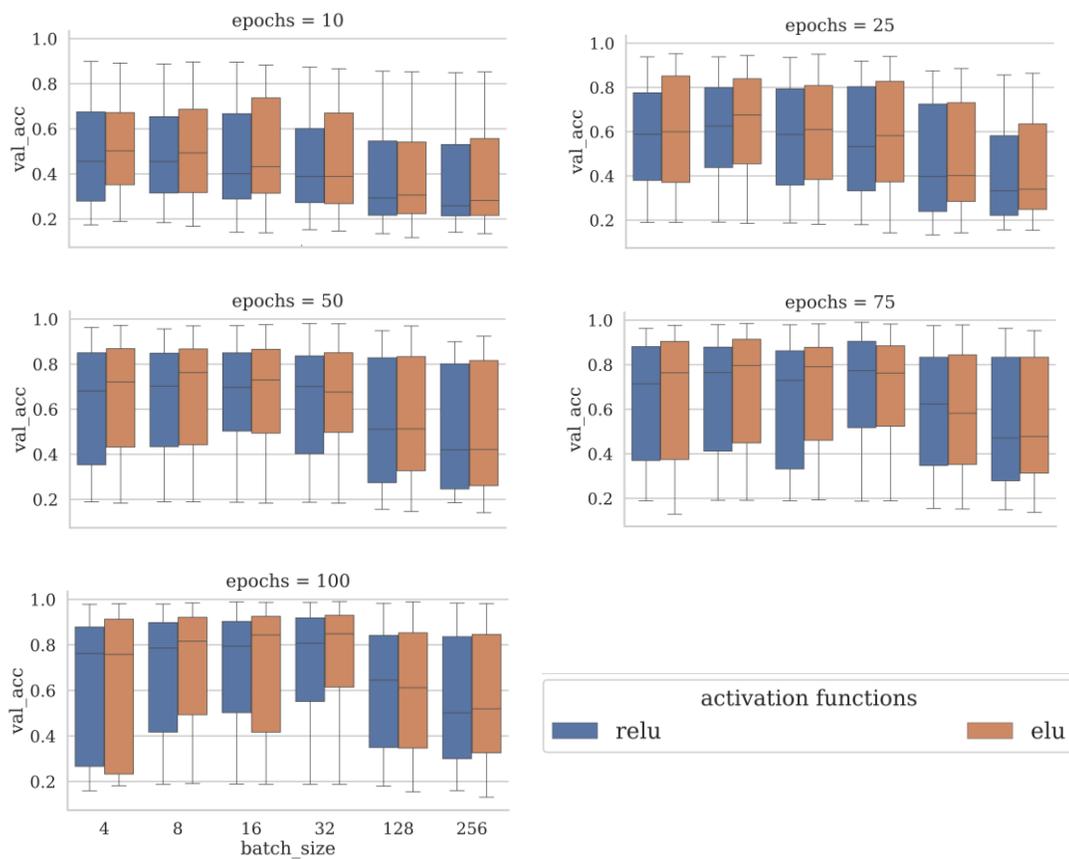


Figure 6.15 Hypermeter optimization study on activation functions, relu and elu, using validation accuracy

Two types of activation functions are common to use for classification problems, including softmax and sigmoid. As shown in Figure 6.16, softmax, regardless of the number of epoch or batch size, works better in all cases. Here, a

brief explanation is provided to justify these results. In general, softmax works better for multiclass or mutually exclusive categories in classification problems. And sigmoid works well for classes that are not mutually exclusive. However, it highly depends on the data one works with. Here, as shown in Figure 6.16, both softmax and sigmoid perform equally well. However, softmax is considered as an activation function for the output layer mainly because of two reasons: first, in this study, multi-class classification is conducted, and second, it represents better results during the optimization process for all epochs and batch sizes.

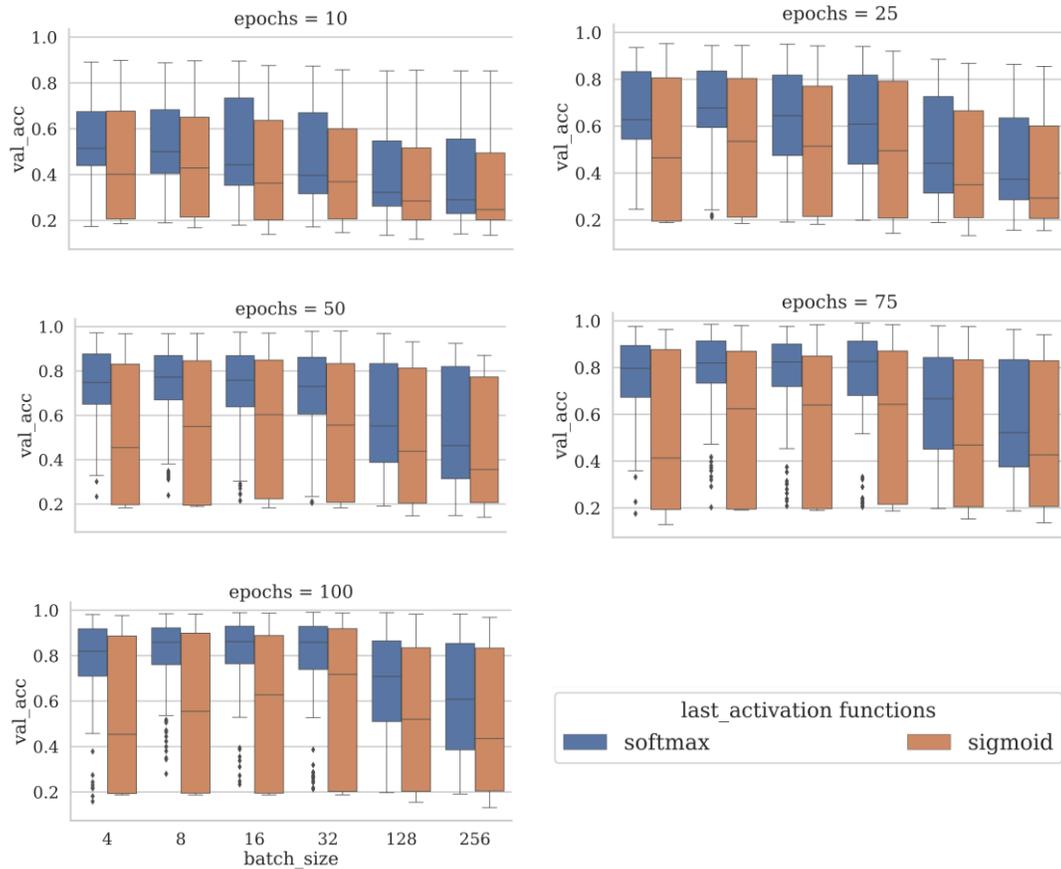


Figure 6.16 Hyperparameter optimization study on last activation functions, sigmoid and softmax, using validation accuracy

6.8 Conclusion

Reducing the limit of detection is addressed in this chapter to allow application of the designed sensor to lower concentrations of materials. It is shown that the versatility of an algorithm is desirable as it allows tuning for high-end applications.

Parameters of the neural network are studied in a broad platform of hyperparameter optimization, including all major parameters such as the number of layers, number of neurons in each layer, activation function, loss-function, and optimizer. This parametric sweep leads to various opportunities to adapt the network to specific applications.

As shown in Figure 6.17(a), the network can reach 95% accuracy at 6 epochs and higher. Similarly, the confusion matrix shown below confirms how accurate input data mapping is for an optimized neural network that guarantees accurate discrimination of concentrations as low as 1% methanol in water from pure water, reducing the limit of detection down to 1%.

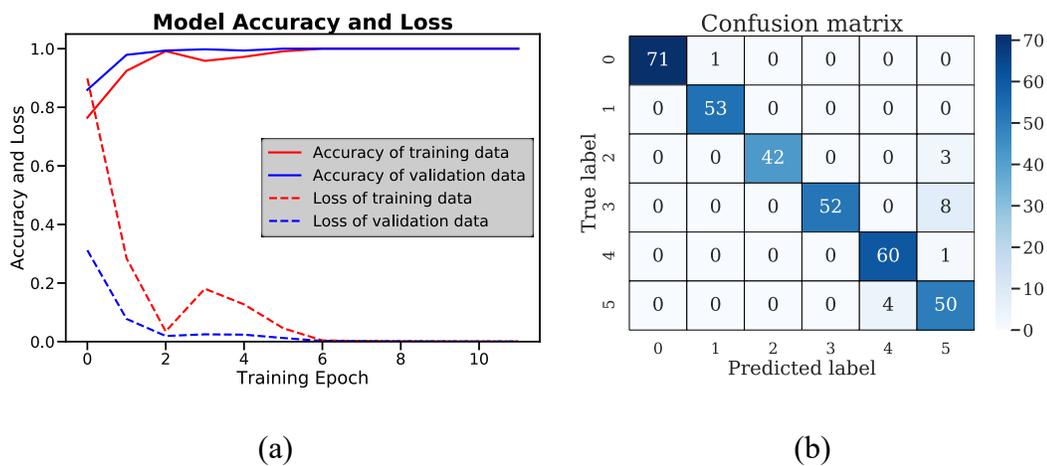


Figure 6.17 (a) Accuracy and loss of training and validation data, (b) Confusion matrix for the optimized neural network resulting in 95 % accuracy.

Chapter 7 Conclusions and Future Work

In this chapter, the main findings of the thesis are summarized, and suggestions for future work are presented.

7.1 Thesis conclusions and contributions

This thesis is concerned with the use of microwave sensors for material characterization and with the impact of environmental error sources (especially temperature) on its response. The conclusions and contributions can be summarized as follows:

- Understanding whether the sensor response is still reliable or not after the undesired effect of temperature on both sensory circuit and material. The sensor response is studied when temperature increases and simulation results are provided as a proof of concept compared with the experiment described in Chapter 3.
- Compensation of temperature effect from the response besides reporting the temperature of the material at the same time is performed using machine learning algorithms. Preparation of the input data fed to the network is explained, and a comparison between meaningful and unsatisfactory input data is provided in Chapter 4.
- Development of different machine-learning algorithms for classification (removing the effect of temperature from the sensor response) and regression (reporting the temperature of the material) is discussed in Chapter 5.

- The use of optimization algorithms to improve the performance of the multi-layer perceptron for better classification in low concentrations of material is explained in Chapter 6.

7.2 Suggestions for future work

There are a number of way this research can be extended and modified:

- A study could be performed on the active circuit to make it stable when the temperature increases, especially for working in a harsh environmental situation when the temperature increases higher than 100°C (Chapter 4).
- Other algorithms, such as autoencoder, could be used as a single algorithm to report both type and temperature of the material simultaneously (Chapter 5).
- Algorithms to remove more than one type of error from the sensor response could be developed (Chapter 5).
- Hyperparameter optimization could be extended to other algorithms, such as logistic regression, rather than using default functions from Keras (Chapter 6).
- In order to further optimize the model instead of searching for the optimizer or loss functions with their default parameters, one can also stick to one optimizer/loss function and delve into its intrinsic parameters (Chapter 6).

References

- [1] M. Abdolrazzaghi, M. Daneshmand, and A. K. Iyer, “Strongly Enhanced Sensitivity in Planar Microwave Sensors Based on Metamaterial Coupling,” *IEEE Trans. Microw. Theory Tech.*, vol. 66, no. 4, pp. 1843–1855, Apr. 2018, doi: 10.1109/TMTT.2018.2791942.
- [2] H. Moghadas, M. Daneshmand, and P. Mousavi, “A passive non-contact microwave loop resonance sensor for liquid interface,” *Sens. Actuators B Chem.*, vol. 241, pp. 96–98, Mar. 2017.
- [3] M. Abdolrazzaghi and M. Daneshmand, “Dual Active Resonator for Dispersion Coefficient Measurement of Asphaltene Nano-Particles,” *IEEE Sens. J.*, vol. 17, no. 22, pp. 7248–7256, Nov. 2017, doi: 10.1109/JSEN.2017.2734692.
- [4] A. Salim and S. Lim, “Complementary Split-Ring Resonator-Loaded Microfluidic Ethanol Chemical Sensor,” *Sensors*, vol. 16, no. 11, Oct. 2016.
- [5] A. Mason, O. Korostynska, M. Ortoneda-Pedrola, A. Shaw, and A. Al-Shamma’A, “A resonant co-planar sensor at microwave frequencies for biomedical applications,” *Sensors Actuators, A Phys.*, vol. 202, pp. 170–175, 2013, doi: 10.1016/j.sna.2013.04.015.
- [6] M. A. H. Ansari, A. K. Jha, and M. J. Akhtar, “Design and Application of the CSRR-Based Planar Sensor for Noninvasive Measurement of Complex Permittivity,” *IEEE Sens. J.*, 2015, doi: 10.1109/JSEN.2015.2469683.
- [7] N. Sharafadinzadeh, M. Abdolrazzaghi, and M. Daneshmand, “Highly sensitive microwave split ring resonator sensor using gap extension for glucose sensing,” in *2017 IEEE MTT-S International Microwave Workshop Series on Advanced Materials and Processes for RF and THz Applications, IMWS-AMP 2017*, 2018, vol. 2018-Janua, doi: 10.1109/IMWS-AMP.2017.8247400.
- [8] H. J. Lee, K. A. Hyun, and H. Il Jung, “A high-Q resonator using biocompatible materials at microwave frequencies,” *Appl. Phys. Lett.*, vol.

References

- 104, no. 2, p. 023509, Jan. 2014, doi: 10.1063/1.4862029.
- [9] M. Abdolrazzaghi, A. Abdolali, and M. Daneshmand, "Highly sensitive miniaturized bio-sensor using 2-layer double split ring resonators," in *2015 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, 2015, pp. 736–737, doi: 10.1109/APS.2015.7304755.
- [10] C.-Y. Hsiao and Y.-C. Chiang, "A miniaturized open-loop resonator filter constructed with floating plate overlays," *Prog. Electromagn. Res. C*, vol. 14, pp. 131–145, 2010, doi: 10.2528/PIERC10051405.
- [11] M. Abdolrazzaghi, A. Abdolali, and S. Hashemy, "Improvements in DNA biosensors using joint split ring resonators coupled with thin film microstrip line," *Appl. Comput. Electromagn. Soc. J.*, 2016.
- [12] Z. Abbasi, P. Shariaty, M. Nosrati, Z. Hashisho, and M. Daneshmand, "Dual-Band Microwave Circuits for Selective Binary Gas Sensing System," *IEEE Trans. Microw. Theory Tech.*, vol. 67, no. 10, pp. 4206–4219, Oct. 2019, doi: 10.1109/TMTT.2019.2934459.
- [13] A. Bogner, C. Steiner, S. Walter, J. Kita, G. Hagen, and R. Moos, "Planar Microstrip Ring Resonators for Microwave-Based Gas Sensing: Design Aspects and Initial Transducers for Humidity and Ammonia Sensing," *Sensors*, vol. 17, no. 10, p. 2422, Oct. 2017, doi: 10.3390/s17102422.
- [14] A. Quddious *et al.*, "Disposable, Paper-Based, Inkjet-Printed Humidity and H₂S Gas Sensor for Passive Sensing Applications," *Sensors*, vol. 16, no. 12, p. 2073, Dec. 2016.
- [15] A. Bababjanyan, H. Melikyan, S. Kim, J. Kim, K. Lee, and B. Friedman, "Real-Time Noninvasive Measurement of Glucose Concentration Using a Microwave Biosensor," *J. Sensors*, vol. 2010, pp. 1–7, 2010, doi: 10.1155/2010/452163.
- [16] A. Ebrahimi, W. Withayachumnankul, S. F. Al-Sarawi, and D. Abbott, "Microwave microfluidic sensor for determination of glucose concentration in water," in *2015 IEEE 15th Mediterranean Microwave Symposium (MMS)*, 2015, pp. 1–3, doi: 10.1109/MMS.2015.7375441.

References

- [17] M. Hofmann, G. Fischer, R. Weigel, and D. Kissinger, “Microwave-based noninvasive concentration measurements for biomedical applications,” *IEEE Trans. Microw. Theory Tech.*, 2013, doi: 10.1109/TMTT.2013.2250516.
- [18] P. Velez, J. Munoz-Enano, K. Grenier, J. Mata-Contreras, D. Dubuc, and F. Martin, “Split Ring Resonator-Based Microwave Fluidic Sensors for Electrolyte Concentration Measurements,” *IEEE Sens. J.*, vol. 19, no. 7, pp. 2562–2569, Apr. 2019, doi: 10.1109/JSEN.2018.2890089.
- [19] T. Chretiennot, D. Dubuc, and K. Grenier, “Double stub resonant biosensor for glucose concentrations quantification of multiple aqueous solutions,” in *2014 IEEE MTT-S International Microwave Symposium (IMS2014)*, 2014, pp. 1–4, doi: 10.1109/MWSYM.2014.6848570.
- [20] W. Buff, “SAW sensors,” *Sensors Actuators A Phys.*, vol. 30, no. 1, pp. 117–121, 1992, doi: 10.1016/0924-4247(92)80205-H.
- [21] K. Länge, B. E. Rapp, and M. Rapp, “Surface acoustic wave biosensors: a review,” *Anal. Bioanal. Chem.*, vol. 391, no. 5, pp. 1509–19, Jul. 2008, doi: 10.1007/s00216-008-1911-5.
- [22] K. Länge, “Bulk and surface acoustic wave sensor arrays for multi-analyte detection: A review,” *Sensors (Switzerland)*, 2019, doi: 10.3390/s19245382.
- [23] A. J. Bandodkar *et al.*, “Re-usable electrochemical glucose sensors integrated into a smartphone platform,” *Biosens. Bioelectron.*, vol. 101, pp. 181–187, Mar. 2018, doi: 10.1016/j.bios.2017.10.019.
- [24] L. Wang *et al.*, “Ratiometric electrochemical glucose sensor based on electroactive Schiff base polymers,” *Sensors Actuators, B Chem.*, 2019, doi: 10.1016/j.snb.2019.01.061.
- [25] K. Y. You, Z. Abbas, M. F. A. Malek, and E. M. Cheng, “Non-destructive Dielectric Measurements and Calibration for Thin Materials Using Waveguide-Coaxial Adaptors,” *Meas. Sci. Rev.*, vol. 14, no. 1, pp. 16–24, Feb. 2014, doi: 10.2478/MSR-2014-0003.
- [26] K. Saeed, R. D. Pollard, and I. C. Hunter, “Substrate Integrated Waveguide Cavity Resonators for Complex Permittivity Characterization of Materials,”

References

- IEEE Trans. Microw. Theory Tech.*, vol. 56, no. 10, pp. 2340–2347, Oct. 2008, doi: 10.1109/TMTT.2008.2003523.
- [27] Q. Tan, Y. Guo, L. Zhang, F. Lu, H. Dong, and J. Xiong, “Substrate Integrated Waveguide (SIW)-Based Wireless Temperature Sensor for Harsh Environments,” *Sensors*, vol. 18, no. 5, p. 1406, May 2018, doi: 10.3390/s18051406.
- [28] A. Salim, S. H. Kim, J. Y. Park, and S. Lim, “Microfluidic Biosensor Based on Microwave Substrate-Integrated Waveguide Cavity Resonator,” *J. Sensors*, 2018, doi: 10.1155/2018/1324145.
- [29] A. K. Jha and M. J. Akhtar, “SIW cavity based RF sensor for dielectric characterization of liquids,” in *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*, 2014, pp. 1–4, doi: 10.1109/CAMA.2014.7003427.
- [30] L. Benkhaoua, M. T. Benhabiles, S. Mouissat, and M. L. Riabi, “Miniaturized Quasi-Lumped Resonator for Dielectric Characterization of Liquid Mixtures,” *IEEE Sens. J.*, vol. 16, no. 6, pp. 1603–1610, Mar. 2016, doi: 10.1109/JSEN.2015.2504601.
- [31] W. Withayachumnankul, K. Jaruwongrungee, A. Tuantranont, C. Fumeaux, and D. Abbott, “Metamaterial-based microfluidic sensor for dielectric characterization,” *Sensors Actuators A Phys.*, vol. 189, pp. 233–237, Jan. 2013, doi: 10.1016/j.sna.2012.10.027.
- [32] A. Ebrahimi, W. Withayachumnankul, S. Al-Sarawi, and D. Abbott, “High-Sensitivity Metamaterial-Inspired Sensor for Microfluidic Dielectric Characterization,” *IEEE Sens. J.*, vol. 14, no. 5, pp. 1345–1351, May 2014, doi: 10.1109/JSEN.2013.2295312.
- [33] M. A. H. Ansari, A. K. Jha, and M. J. Akhtar, “Dual band microwave sensor for dielectric characterization of dispersive materials,” *Asia-Pacific Microw. Conf. Proceedings, APMC*, vol. 1, pp. 6–8, 2016, doi: 10.1109/APMC.2015.7411595.
- [34] M. Abdolrazzaghi and M. Daneshmand, “A Phase-Noise Reduced Microwave Oscillator Sensor With Enhanced Limit of Detection Using

References

- Active Filter,” *IEEE Microw. Wirel. Components Lett.*, vol. 28, no. 9, pp. 837–839, 2018, doi: 10.1109/LMWC.2018.2850283.
- [35] M. Abdolrazzaghi, M. H. Zarifi, W. Pedrycz, and M. Daneshmand, “Robust Ultra-High Resolution Microwave Planar Sensor Using Fuzzy Neural Network Approach,” *IEEE Sens. J.*, vol. 17, no. 2, pp. 323–332, Jan. 2017, doi: 10.1109/JSEN.2016.2631618.
- [36] M. Abdolrazzaghi, M. H. Zarifi, and M. Daneshmand, “Wireless Communication in Feedback-Assisted Active Sensors,” *IEEE Sens. J.*, vol. 16, no. 22, 2016, doi: 10.1109/JSEN.2016.2604855.
- [37] M. Abdolrazzaghi, F. Hariri, M. Chu, H. Naguib, and M. Daneshmand, “Relative Humidity Sensing using PANI/PVA integrated with Feedback Oscillator Circuit,” in *2019 IEEE SENSORS*, 2019, pp. 1–4, doi: 10.1109/SENSORS43011.2019.8956563.
- [38] M. Abdolrazzaghi and M. Daneshmand, “A 4 GHz Near-Field Monitoring Planar Oscillator Sensor,” in *2018 IEEE MTT-S International Microwave Workshop Series on Advanced Materials and Processes for RF and THz Applications (IMWS-AMP)*, 2018, pp. 1–3, doi: 10.1109/IMWS-AMP.2018.8457169.
- [39] M. Abdolrazzaghi and M. Daneshmand, “Multifunctional Ultrahigh Sensitive Microwave Planar Sensor to Monitor Mechanical Motion: Rotation, Displacement, and Stretch,” *Sensors*, vol. 20, no. 4, p. 1184, Feb. 2020, doi: 10.3390/s20041184.
- [40] M. Abdolrazzaghi, “Advances in Active Resonator based Planar Microwave Sensors for Material Characterization,” University of Alberta, 2017.
- [41] M. Abdolrazzaghi and M. Daneshmand, “Compelling impact of intermodulation products of regenerative active resonators on sensitivity,” in *2017 IEEE MTT-S International Microwave Symposium (IMS)*, 2017, pp. 1018–1021, doi: 10.1109/MWSYM.2017.8058764.
- [42] M. Abdolrazzaghi, S. Hashemy, and A. Abdolali, “Fast-forward solver for inhomogeneous media using machine learning methods: artificial neural network, support vector machine and fuzzy logic,” *Neural Comput. Appl.*,

References

- Nov. 2016, doi: 10.1007/s00521-016-2694-9.
- [43] A. W. Kraszewski and S. O. Nelson, "Observations on resonant cavity perturbation by dielectric objects," *ARS Repr. Collect.*, 1992.
- [44] A. W. Kraszewski and S. O. Nelson, "Resonant-cavity perturbation measurement for mass determination of the perturbing object," in *Conference Proceedings. 10th Anniversary. IMTC/94. Advanced Technologies in I & M. 1994 IEEE Instrumentation and Measurement Technolgy Conference (Cat. No.94CH3424-9)*, pp. 1261–1264, doi: 10.1109/IMTC.1994.351826.
- [45] A. A. Mohd Bahar, Z. Zakaria, M. K. Md. Arshad, A. A. M. Isa, Y. Dasril, and R. A. Alahnomi, "Real Time Microwave Biochemical Sensor Based on Circular SIW Approach for Aqueous Dielectric Detection," *Sci. Rep.*, vol. 9, no. 1, p. 5467, 2019, doi: 10.1038/s41598-019-41702-3.
- [46] X. Yi, C. Cho, J. Cooper, Y. Wang, M. M. Tentzeris, and R. T. Leon, "Passive wireless antenna sensor for strain and crack sensing - Electromagnetic modeling, simulation, and testing," *Smart Mater. Struct.*, 2013, doi: 10.1088/0964-1726/22/8/085009.
- [47] M. Abdolrazzaghi, S. Khan, and M. Daneshmand, "A Dual-Mode Split-Ring Resonator to Eliminate Relative Humidity Impact," *IEEE Microw. Wirel. Components Lett.*, vol. 28, no. 10, pp. 939–941, Oct. 2018, doi: 10.1109/LMWC.2018.2860596.
- [48] M. Abdolrazzaghi and M. Daneshmand, "Enhanced Q double resonant active sensor for humidity and moisture effect elimination," in *IEEE MTT-S International Microwave Symposium Digest*, 2016, vol. 2016-Augus, pp. 1–3, doi: 10.1109/MWSYM.2016.7540006.
- [49] A. Zarnani, S. Karimi, and P. Musilek, "Quantile Regression and Clustering Models of Prediction Intervals for Weather Forecasts: A Comparative Study," *Forecasting*, 2019, doi: 10.3390/forecast1010012.
- [50] T. Barton and P. Musilek, "Day-Ahead Dynamic Thermal Line Rating Using Numerical Weather Prediction," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering, CCECE 2019*, 2019, doi:

References

- 10.1109/CCECE.2019.8861883.
- [51] D. Mocrii, Y. Chen, and P. Musilek, "IoT-based smart homes: A review of system architecture, software, communications, privacy and security," *Internet of Things*, 2018, doi: 10.1016/j.iot.2018.08.009.
- [52] M. Bardwell, J. Wong, S. Zhang, and P. Musilek, "IoT-based MPPT Controller for Photovoltaic Array," in *2018 IEEE Electrical Power and Energy Conference, EPEC 2018*, 2018, doi: 10.1109/EPEC.2018.8598404.
- [53] M. Bardwell, J. Wong, S. Zhang, and P. Musilek, "Design considerations for iot-based pv charge controllers," in *Proceedings - 2018 IEEE World Congress on Services, SERVICES 2018*, 2018, doi: 10.1109/SERVICES.2018.00043.
- [54] J. Konecny, M. Prauzek, M. Borova, K. Janosova, and P. Musilek, "A Simulation Framework for Energy Harvesting in Wireless Sensor Networks: Single Node Architecture Perspective," in *Proceedings of the 23rd International Conference Electronics 2019, ELECTRONICS 2019*, 2019, doi: 10.1109/ELECTRONICS.2019.8765580.
- [55] B. Lashkari, Y. Chen, and P. Musilek, "Energy management for smart homes-state of the art," *Applied Sciences (Switzerland)*. 2019, doi: 10.3390/app9173459.
- [56] M. Prauzek, N. R. A. Mourcet, J. Hlavica, and P. Musilek, "Q-Learning Algorithm for Energy Management in Solar Powered Embedded Monitoring Systems," in *2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings*, 2018, doi: 10.1109/CEC.2018.8477781.
- [57] Z. Abbasi, H. Niazi, M. Abdolrazzaghi, W. Chen, and M. Daneshmand, "Monitoring pH Level Using High-Resolution Microwave Sensor for Mitigation of Stress Corrosion in Steel Pipelines," *IEEE Sens. J.*, pp. 1–1, 2020, doi: 10.1109/JSEN.2020.2978086.
- [58] Z. Abbasi, M. Baghelani, and M. Daneshmand, "Zero Power Consumption Chipless Distant Microwave Moisture Sensor for Smart Home Applications," in *Proceedings of IEEE Sensors*, 2019, doi: 10.1109/SENSORS43011.2019.8956753.

References

- [59] Z. Abbasi, M. Baghelani, M. Nosrati, A. Sanati-Nezhad, and M. Daneshmand, "Real-Time Non-Contact Integrated Chipless RF Sensor for Disposable Microfluidic Applications," *IEEE J. Electromagn. RF Microwaves Med. Biol.*, 2019, doi: 10.1109/jerm.2019.2954219.
- [60] N. Hosseini, M. Baghelani, and M. Daneshmand, "Selective Volume Fraction Sensing Using Resonant-Based Microwave Sensor and Its Harmonics," *IEEE Trans. Microw. Theory Tech.*, 2020, doi: 10.1109/tmtt.2020.2990139.
- [61] N. Hosseini, S. S. Olokede, and M. Daneshmand, "A novel miniaturized asymmetric CPW split ring resonator with extended field distribution pattern for sensing applications," *Sensors Actuators, A Phys.*, 2020, doi: 10.1016/j.sna.2019.111769.
- [62] Y.-H. Kim, K. Jang, Y. J. Yoon, and Y.-J. Kim, "A novel relative humidity sensor based on microwave resonators and a customized polymeric film," *Sensors Actuators B Chem.*, vol. 117, no. 2, pp. 315–322, Oct. 2006, doi: 10.1016/j.snb.2005.11.004.
- [63] H. Yu *et al.*, "Design and analysis of ultrafast and high-sensitivity microwave transduction humidity sensor based on belt-shaped MoO₃ nanomaterial," *Sensors Actuators, B Chem.*, 2020, doi: 10.1016/j.snb.2019.127138.
- [64] P. Vélez, J. Muñoz-Enano, M. Gil, J. Mata-Contreras, and F. Martín, "Differential microfluidic sensors based on dumbbell-shaped defect ground structures in microstrip technology: Analysis, optimization, and applications," *Sensors (Switzerland)*, 2019, doi: 10.3390/s19143189.
- [65] A. Ebrahimi, J. Scott, and K. Ghorbani, "Transmission lines terminated with LC resonators for differential permittivity sensing," *IEEE Microw. Wirel. Components Lett.*, 2018, doi: 10.1109/LMWC.2018.2875996.
- [66] J. Muñoz-Enano, P. Vélez, M. Gil, and F. Martín, "Microfluidic reflective-mode differential sensor based on open split ring resonators (OSRRs)," *Int. J. Microw. Wirel. Technol.*, 2020, doi: 10.1017/s1759078720000501.
- [67] A. A. Abduljabar, N. Clark, J. Lees, and A. Porch, "Dual Mode Microwave

References

- Microfluidic Sensor for Temperature Variant Liquid Characterization,” *IEEE Trans. Microw. Theory Tech.*, vol. 65, no. 7, pp. 2572–2582, Jul. 2017, doi: 10.1109/TMTT.2016.2647249.
- [68] A. A. Abduljabar, H. Hamzah, and A. Porch, “Double Microstrip Microfluidic Sensor for Temperature Correction of Liquid Characterization,” *IEEE Microw. Wirel. Components Lett.*, vol. 28, no. 8, pp. 735–737, Aug. 2018, doi: 10.1109/LMWC.2018.2849218.
- [69] H. R. Sun *et al.*, “Symmetric coplanar waveguide sensor loaded with interdigital capacitor for permittivity characterization,” *Int. J. RF Microw. Comput. Eng.*, 2020, doi: 10.1002/mmce.22023.
- [70] L. Harrsion, M. Ravan, D. Tandel, K. Zhang, T. Patel, and R. K. Amineh, “Material identification using a microwave sensor array and machine learning,” *Electron.*, 2020, doi: 10.3390/electronics9020288.
- [71] M. Abdolrazzaghi, N. Kazemi, and M. Daneshmand, “Sensitive Spectroscopy Using DSRR Array and Linvill Negative Impedance,” in *2019 IEEE MTT-S International Microwave Symposium (IMS)*, 2019, pp. 1080–1083, doi: 10.1109/MWSYM.2019.8701104.
- [72] H.-Y. Gan *et al.*, “Differential Microwave Microfluidic Sensor Based on Microstrip Complementary Split-Ring Resonator (MCSRR) Structure,” *IEEE Sens. J.*, 2020, doi: 10.1109/jsen.2020.2973196.
- [73] P. Wang and A. Anderko, “Computation of dielectric constants of solvent mixtures and electrolyte solutions,” *Fluid Phase Equilib.*, 2001, doi: 10.1016/S0378-3812(01)00507-6.
- [74] F. Carpi, G. Gallone, F. Galantini, and D. De Rossi, “Enhancing the dielectric permittivity of elastomers,” in *Dielectric Elastomers as Electromechanical Transducers*, Elsevier, 2008, pp. 51–68.
- [75] E. Tuncer, S. M. Gubański, and B. Nettelblad, “Dielectric relaxation in dielectric mixtures: Application of the finite element method and its comparison with dielectric mixture formulas,” *J. Appl. Phys.*, vol. 89, no. 12, pp. 8092–8100, Jun. 2001, doi: 10.1063/1.1372363.
- [76] J. C. M. Garnett and J. Larmor, “Colours in metal glasses and in metallic

References

- films.,” *Proc. R. Soc. London*, vol. 73, no. 488–496, pp. 443–445, Jul. 1904, doi: 10.1098/rspl.1904.0058.
- [77] J. C. M. Garnett, “VII. Colours in metal glasses, in metallic films, and in metallic solutions.—II,” *Philos. Trans. R. Soc. London. Ser. A, Contain. Pap. a Math. or Phys. Character*, vol. 205, no. 387–401, pp. 237–288, Jan. 1906, doi: 10.1098/rsta.1906.0007.
- [78] V. M. Shalaev, “Electromagnetic properties of small-particle composites,” *Physics Report*. 1996, doi: 10.1016/0370-1573(95)00076-3.
- [79] G. B. Smith, “Dielectric constants for mixed media,” *J. Phys. D. Appl. Phys.*, vol. 10, no. 4, pp. L39–L42, Mar. 1977, doi: 10.1088/0022-3727/10/4/004.
- [80] A. Andryieuski, S. M. Kuznetsova, S. V. Zhukovsky, Y. S. Kivshar, and A. V. Lavrinenko, “Water: Promising Opportunities For Tunable All-dielectric Electromagnetic Metamaterials,” *Sci. Rep.*, vol. 5, no. 1, p. 13535, Oct. 2015, doi: 10.1038/srep13535.
- [81] M. Onimisi, J. Ikyumbur, S. Abdu, and E. Hembra, “Frequency and Temperature Effect on Dielectric Properties of Acetone and Dimethylformamide,” *Phys. Sci. Int. J.*, 2016, doi: 10.9734/psij/2016/27742.
- [82] A. P. G. and R. N. Clarke, “Tables of the Complex Permittivity of Dielectric Reference Liquids at Frequencies up to 5GHz,” *NPL Rep. MAT23*, 2012, doi: 10.1007/s13398-014-0173-7.2.
- [83] Y. Li *et al.*, “Separate wind power and ramp predictions based on meteorological variables and clustering method,” in *2016 IEEE 6th International Conference on Power Systems, ICPS 2016*, 2016, doi: 10.1109/ICPES.2016.7584025.
- [84] P. Musilek, P. Kromer, and T. Barton, “E-BACH: Entropy-based clustering hierarchy for wireless sensor networks,” in *Proceedings - 2015 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015*, 2016, doi: 10.1109/WI-IAT.2015.88.
- [85] “EL-WiFi-TH.” .
- [86] G. Yin, “Stochastic Approximation and Its Applications,” *J. Am. Stat.*

References

- Assoc.*, 2004, doi: 10.1198/jasa.2004.s319.
- [87] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *Ann. Math. Stat.*, 1951, doi: 10.1214/aoms/1177729586.
- [88] Y. Yao, L. Rosasco, and A. Caponnetto, “On early stopping in gradient descent learning,” *Constr. Approx.*, 2007, doi: 10.1007/s00365-006-0663-2.
- [89] G. E. Hinton, “Optimization: How to make the learning go faster,” *Coursera*, 2012, doi: <https://www.coursera.org/learn/neural-networks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude>.
- [90] U. Michelucci and U. Michelucci, “Training Neural Networks,” in *Applied Deep Learning*, 2018.
- [91] G. E. Hinton, N. Srivastava, and K. Swersky, “Lecture 6.5- Divide the gradient by a running average of its recent magnitude,” in *COURSERA: Neural Networks for Machine Learning*, 2012.
- [92] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [93] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, “A sufficient condition for convergences of adam and rmsprop,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, doi: 10.1109/CVPR.2019.01138.
- [94] T. Dozat, “Incorporating Nesterov Momentum into Adam,” *ICLR Work.*, 2016.
- [95] R. Gylberth, R. Adnan, S. Yazid, and T. Basaruddin, “Differentially private optimization algorithms for deep neural networks,” in *2017 International Conference on Advanced Computer Science and Information Systems, ICACISIS 2017*, 2018, doi: 10.1109/ICACISIS.2017.8355063.
- [96] S. Indrapriyadarsini, S. Mahboubi, H. Ninomiya, and H. Asai, “A Stochastic Quasi-Newton Method with Nesterov’s Accelerated Gradient,” 2020.
- [97] “Optimizing Gradient Descent.” .
- [98] A. L. Caterini and D. E. Chang, “Recurrent neural networks,” in

References

- SpringerBriefs in Computer Science*, 2018.
- [99] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Journal of Machine Learning Research*, 2010.
- [100] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2012, doi: 10.1007/978-3-642-35289-8-26.
- [101] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,” in *32nd International Conference on Machine Learning, ICML 2015*, 2015.
- [102] L. Deng, G. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: An overview,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013, doi: 10.1109/ICASSP.2013.6639344.
- [103] S. Elfving, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” *Neural Networks*, 2018, doi: 10.1016/j.neunet.2017.12.012.
- [104] F. Ertam, “Data classification with deep learning using tensorflow,” in *2nd International Conference on Computer Science and Engineering, UBMK 2017*, 2017, doi: 10.1109/UBMK.2017.8093521.
- [105] “Searching for activation functions,” in *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018.
- [106] L. Mou, P. Ghamisi, and X. X. Zhu, “Deep recurrent neural networks for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, 2017, doi: 10.1109/TGRS.2016.2636241.