Sensor Qualification In An Industrial Environment Using A Convolutional Neural Network

Daniel Henrique da Costa Sousa

Thesis to obtain the Master of Science Degree in

Engineering Physics

Supervisor(s): Prof.ª Doutora Susana Isabel Pinheiro Cardoso de Freitas

Examination Committee

Chairperson: Prof. Doutor Carlos Manuel dos Santos Rodrigues da Cruz
Supervisor: Prof.ª Doutora Susana Isabel Pinheiro Cardoso de Freitas
Member of the Committee: Doutora Ana Neves Vieira da Silva

December 2020

Out of the cradle
onto dry land
here it is standing:
atoms with consciousness;
matter with curiosity.
Richard P. Feynman

Acknowledgments

Foremost I would like to thank Professor Susana Freitas for her guidance and the opportunity given to work at Instituto de Engenharia de Sistemas e Computadores – Microsistemas e Nanotecnologias. Without Professor Susana and the courses on nanotechnology taught at Instituto Superior Técnico, I cannot be sure that this would be the path that I would follow. I would also like to thank the process engineers for their help and advice on good practices to be held in the cleanroom.

Now it is time to thank my colleagues/friends who without I am not sure I would finish this work on time or even if I would be able to finish it at all. Mafalda Ferreira and Pedro Araújo were crucial in this matter, and I am forever in their debt. I want to thank Professor Ana Silva and Tiago Coutinho, who always gave me tips and pieces of advice on my work and were the first users of the code developed. Their help was crucial to understand some of the future requirements of users.

As I feel that this marks the end of a cycle in my life, I want to express my eternal gratitude to every colleague and professor with whom I have shared moments during these five years. They are too many to name, but a special thank you note goes to Beatriz and Rodrigo.

I want to thank my family. My mom had a crucial role in supporting me during these five years. My brother always tried to help me with anything that he could and always provided some distraction and a good laugh. My dad and grandfather who were very interested in my work even though not having any knowledge in this area.

Finally, I want to thank everyone that will spend their time reading this work; I hope that you find it as engaging and exciting as I did.

Resumo

Atualmente os sensores tem vindo a ganhar importância nas nossas vidas. Os sensores Magnetoresistivos tem provado o seu valor em muitas aplicações, desde a indústria automóvel a aplicações médicas. Como tal, existe uma constante necessidade de otimizar estes dispositivos e o rendimento da sua produção.

Em qualquer processo de nano/microfabricação, o tempo é um dos recursos mais valiosos. É crucial existirem ferramentas que permitam avaliar rapidamente se as estruturas produzidas estão de acordo com o esperado.

Neste trabalho é apresentado um programa que permite a análise quantitativa de alguns parâmetros críticos na produção de dispositivos Magnetoresistivos. No *software* desenvolvido, uma Rede Neuronal Convolucional é utilizada com a finalidade de ajudar os utilizadores a obter uma classificação autónoma de dispositivos magnetoresistivos, com foco em sensores. Além do *software* para análise de parâmetros quantitativos assim como o classificador de curvas para sensores, é também discutido o uso dum algoritmo de aprendizagem automática para prever o rendimento de um dado processo de nano/microfabricação.

Palavras-chave: Sensores Magnetoresistivos, Microfabricação, Qualificação de Sensores, Aprendizagem Automática, Rede Neuronal Convolucional

Abstract

Nowadays, sensors occupy a significant role in our everyday lives. Magnetoresistive sensors have been proven to be very useful in many applications, from sensing magnetic fields to precision encoding. As a consequence, there is a constant demand to optimize such devices and maximize their production yield.

In any nano/microfabrication process, time is one of the most valuable resources. It is crucial to have tools to evaluate rapidly if the produced structures are within the expected outcomes or not.

This work presents a tool for quantitative analysis of some critical parameters of Magnetoresistive Devices fabrication. Furthermore, the software includes an automatic curve classifier using a Convolutional Neural Network to qualify Magnetoresistive Devices, focusing on sensors. Alongside the software for quantitative analysis and the curve classifier for sensors, it is also briefly discussed the use of a machine learning-based algorithm to predict the yield of a given nano/microfabrication fabrication process.

Keywords: Magnetoresistive Sensors, Microfabrication, Sensor Qualification, Machine Learning, Convolutional Neural Network

Contents

List of Tables

List of Figures

Nomenclature

μ_0	Permeability of Free Space
χ	Magnetic Susceptibility
$\vec{B}$	Magnetic Induction Field
$\vec{H}$	Magnetic Field
$\vec{M}$	Magnetization
$\vec{m}$	Net Magnetic Moment
A	Area
CCE	Categorical Cross-Entropy
FN	False Negatives
FP	False Positives
H_c	Magnetic Coercivity
H_{Sat}	Magnetic Saturation Field
M_s	Saturation Magnetization
MR	Magnetoresistance
R	Resistance
r	Radius
R_{High}	High Resistance State
R_{Low}	Low Resistance State
R_{min}	Minimum of Resistance
S	Sensitivity
t	Thickness
TN	True Negatives

TP	True Positives
V	Volume
#	Number
AMR	Anisotropic Magnetoresistance
ANN	Artificial Neural Network
ARDE	Aspect Dependent Etch Rate
CNN	Convolutional Neural Network
CVD	Chemical Vapor Deposition
DC	Direct Current
DWL	Direct Write Laser
EBL	Electron Beam Lithography
EUV	Extreme Ultraviolet Lithography
FL	Free Layer
FM	Ferromagnetic
GMR	Giant Magnetoresistance
GPU	Graphics Processing Unit
IBD	Ion Beam Deposition
INESC MN	Instituto de Engenharia de Sistemas de Computadores - Microsistemas e Nanotecnologias
MR	Magnetoresistive
MTJ	Magnetic Tunnel Junction
NM	Non-magnetic
OC	Open Circuit
PR	Photoresist
PVD	Physical Vapor Deposition
RF	Radio Frequency
RL	Reference Layer
SC	Short Circuit
SV	Spin Valve
TMR	Tunneling Magnetoresistance

Chapter 1

Introduction

1.1 Motivation

Nowadays sensoric electronics importance is irrefutable. There is a constant demand to push forward limits in matters of detection, spatial resolution, sensor foot print, power consumption, etc. Ref. [1]

In applications that need sensing magnetic fields such as biomedical Refs. [2, 3], smart cities in Refs. [4, 5], magnetoresistive (MR) sensors have attracted much interest because of their high sensitivity, low power consumption, low cost and small size, Refs. [1, 6, 7]. Besides being a mature technology, magnetoresistive sensorics are an attractive topic currently pursued by companies toward front end technology. Refs. [8, 9]

Most of these electronics are built using silicon-based semiconductor processes, where the device is built using a top-down approach. There are mainly three types of processes that can be used to build the majority of devices, and these are: (i) Deposition of materials (films); (ii) Lithography; (iii) Etching.

Depending on the dimensions of the final device, one can use micro or nanofabrication techniques. All techniques have some execution challenges, of diverse nature. Ref. [10]

Due to the typical sizes of the elements involved, tens of microns down to a few nanometers, strict micro/nanometric control is essential in all of the fabrication processes. Fluctuations in the process conditions can compromise the behavior of the final device, either electrically (e.g., Resistance Area product, $R \times A$) or magnetically (e.g., saturation magnetization, M_S).

Metrology is then critical for process validation and control specially evolving multilevel steps.

Quality assessment can be time-consuming; one may need to do quality control using many different techniques. One common technique to test produced MR sensors is by measuring their resistance (R) across a varying external magnetic field (H), $R(H)$ is one of the main figures of merit of these devices. In an eight-inch wafer there are tens of thousands of sensors, corresponding to tens of thousands of $R(H)$ sets of points for each sensor. To fully qualify the produced wafer, one would have to manually classify every sensor, spending a considerable amount of time evaluating $R(H)$ loops. Additionally the quality of the sensors cannot be tackled only by evaluating easily defined numerical parameters. The shape of the loop is also relevant, hence using machine learning algorithms to help in the classification

is extremely profitable. If it takes 5 s to classify each of the tens of thousands of the $R(H)$ loops on a eight-inch wafer, then one would take at least one day to qualify it completely. A faster solution is, therefore, needed.

Another constant challenge that exists at micro and nanofabrication facilities, namely at *Instituto de Engenharia de Sistemas e Computadores - Microsistemas e Nanotecnologias* (INESC MN), is to improve the yield of manufacturing processes. By having the highest yield possible, one maximizes the use of time, resources, and hence money.

1.2 Objectives

In this Master Thesis, there are three different objectives:

1. Build an easy to use software that can digest ".SMP" files and provide easy visualization of many important parameters for MR devices as e.g. $R(H)$ plots, the minimal resistance, the magnetoresistance percentage (MR), among others;

2. Build a machine learning-based tool to facilitate the analysis of large volumes of data, regarding $R(H)$ measurements of sensors, while providing an automatic classification of the $R(H)$ hysteresis curve shape;

3. Propose a method to predict the yield of a micro or nanofabrication process. The method should allow the identification of the critical steps of the manufacturing process to improve the overall yield.

1.3 Topic Overview

Currently, at INESC MN, there is already a experimental setup that allows measuring $R(H)$ loops for different sensors across a wafer. Associated to that setup there is a software that returns all the relevant quantities for the analysis stored in a file of ".SMP" extension.

There are already a few student codes that allow to analyze the ".SMP" files and retrieve critical physical quantities such as sensibility, coercivity, etc. There is not, however, one that allows easy visualization of these results. None of the existing codes have a graphical user interface, for instance. There is currently no automated way to classify MR sensors; usually, the users have to manually evaluate $R(H)$ plots to classify them, which is time-consuming.

1.4 Thesis Outline

This Thesis has five chapters and two appendices. They cover the following topics:

- **Chapter 2** - Introduces all the necessary theoretical and state-of-the-art topics in order to contextualize and help understand the work developed in this master thesis;
- **Chapter 3** - Description/results of the work done for the data processor, sensor classifier and data visualizer;
- **Chapter 4** - Proposal of an approach towards predicting the yield of a micro or nanofabrication process for the manufacturing of MR sensors;
- **Chapter 5** - Conclusion and future work, which would further increase the relevance of this work;
- **Appendix A** - Relevant code that used in this Thesis;
- **Appendix B** - Process Characterization Runsheet proposed to be implemented to gather data towards training a yield predictor.

At the beginning of each chapter, one will find a brief introduction to it.

Chapter 2

State-of-the-Art

In this chapter, the State-of-the-Art of topics relevant to a better understating of this master thesis are discussed.

2.1 Theoretical Overview

In this section, critical theoretical concepts for this work are briefly discussed and reviewed.

2.1.1 Magnetoresistance

In 1857 a novel and fascinating phenomenon was discovered and published by Lord Kelvin, Ref. [11]. In this novel experiment published by *The Royal Society of London*, the author described the following: ”(...) I found that iron, when subjected to magnetic force, acquires an increase of resistance to the conduction of electricity along, and a diminution of resistance to the conduction of electricity across, the lines of magnetization*. By experiments more recently made, I have ascertained that the electric conductivity of nickel is similarly influenced by magnetism, but to a greater degree, and with a curious difference from iron in the relative magnitudes of the transverse and longitudinal effects. (...)”.

Although the phenomenon that some material's resistance is dependent on the magnetic field applied was discovered in the middle of the 19th century, it was only at the end of the 20th century when Albert Fert, Ref. [12] and Peter Grünberg, Ref. [13], described both the Giant Magnetoresistance (GMR) effect independently, that this effect started to be used by the industry. This phenomenon gave rise to an emerging scientific/technologic field named Spintronics. This discovery was awarded the Nobel Prize in Physics of 2007, Ref. [14].

To characterize such devices, one important quantity is named after the effect. The MR is defined as:

$$MR = \frac{R_{High} - R_{Low}}{R_{Low}} \tag{2.1}$$

In the Eq. 2.1, R_{High} is the high state resistance and R_{Low} the low state resistance.

Both groups noticed that the resistance of two adjacent ferromagnetic (FM) layers separated by a

non-magnetic (NM) layer, typically referred to as spacer, is dependent on the angle between magnetizations and the external magnetic field applied.

Physically the GMR described by A. Fert, and P. Grünberg is a consequence of the asymmetry in the diffusive scattering of the conductions electrons at NM/FM interfaces.

Depending on the type of the spacer, different phenomena occur, Ref. [15]. The most relevant types of spacers/phenomena are:

- If the spacer is a conductive NM layer, this effect gains the name Giant Magnetoresistance (GMR). Refs. [12, 13];
- In the case of a thin insulating spacer such as an oxide, where the electrons can pass from one FM layer to the other by quantum tunneling effect, the phenomena is named Tunneling Magnetoresistance (TMR). Refs. [16, 17] In this case the Spin-Depedent Tunneling is the mechanism behind the TMR.

There are other types of devices/phenomena, such as the Anisotropic Magnetoresistance (AMR), which is a single layer type of structure but with lower values of MR and smaller linear range.

This Thesis will focus on GMR and TMR sensors since they are the most recent technologies and have the highest MR values. Most of the samples that I studied were TMR based and hence the Spin-Dependent Tunneling will be addressed on this work.

2.1.1.1 Spin-Dependent Tunneling

MTJs sensors consist of two conducting electrodes separated by a very thin insulating layer, typically ranging from 5-30 Å; electrons may move from one electrode to the other by quantum tunneling. Although the electrons cannot exist in the insulating barrier itself, they can still pass from one ferromagnetic layer to another due to their wave-like behavior. After applying a certain voltage V, a narrow energy band of electrons, eV, around the Fermi energy participates in the transport; these electrons can tunnel between the two ferromagnetic layers, the electrons in the filled state of the first layer can tunnel to the second's layer unoccupied stated.

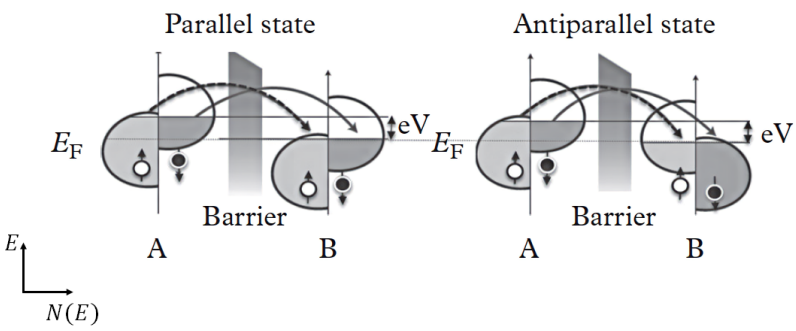


Figure 2.1: Electron energy *vs.* density of states ($N(E)$) for a parallel and antiparallel magnetization state of the ferromagnetic layers. Adapted from Ref. [15], the frame of reference was added. ©2016 Oxford University Press.

In ferromagnetic materials, the electronic d bands are spin-split, meaning that depending on the spin of the electrons, different wave vectors represent them, hence the separation in Fig. 2.1.

The relationship between the Spin-Dependent Tunneling and TMR can be explained by making two assumptions. Ref. [18]

First, let us assume that the electron's spin is conserved during the tunneling; it is easier for the electrons to tunnel through the insulator if both ferromagnetic materials' magnetization is parallel. This assumption about the spin conservation implies that the tunneling of up and down-spin electrons are two independent processes, hence giving rise to two distinct contributions to the electrical current, the contribution from the up-spin electrons ($I_{\uparrow}$) and the contribution by the down-spin electrons ($I_{\downarrow}$).

Consider the parallel state, for the sake of argument let us consider both A and B ferromagnetic layers have a parallel magnetization being that magnetization oriented upwards ($\uparrow\uparrow$). In this case, after applying the voltage across the junction, it is easy for most electrons (down-spin) to find free states for tunneling to the other barrier.

On the other hand, when A and B have an antiparallel configuration ($\uparrow\downarrow$), the situation is different. In this case, the number of free states for most tunneling electrons (down-spin) is small, and hence the higher resistance state.

The second assumption is that the current for each of the spins is proportional to the product of the tunneling density of states of both ferromagnetic layers. In this sense, the following can be written:

$$I_{\uparrow\uparrow} \propto N_A^{\uparrow} N_B^{\uparrow} + N_A^{\downarrow} N_B^{\downarrow}$$

$$I_{\uparrow\downarrow} \propto N_A^{\uparrow} N_B^{\downarrow} + N_A^{\downarrow} N_B^{\uparrow}$$

In the equations above the subscript represents the ferromagnetic layer and the superscript the spin of the electrons. By looking at the equations above and Fig. 2.1, it is evident that the current in the parallel state is different from the antiparallel state, hence giving rise to different values of resistances, assuming, of course, a constant voltage applied.

2.1.2 Magnetization, Magnetic Field and Magnetic Induction

One significant vector quantity to be defined in magnetic materials is the magnetization, $\vec{M}$. It is determined for a sample as the vector sum of its all magnetic moments divided by the sample's volume, Eq. 2.2.

$$\vec{M} = \frac{\vec{m}}{V} \tag{2.2}$$

In Eq. 2.2, $\vec{M}$ is the magnetization, V the volume of the sample, and $\vec{m}$ is the sample's net magnetic moment.

To denote the magnetic induction, also called flux density, one uses $\vec{B}$ and $\vec{H}$ for the magnetic field.

Charges in movement produce magnetic fields, $\vec{H}$. In its essence, the magnetic induction represents a medium's response when subjected to a magnetic field, $\vec{B}$. In short:

$$\vec{B} = \mu_0 \left(\vec{H} + \vec{M} \right) \tag{2.3}$$

In Eq. 2.3, $\vec{M}$ is the magnetization of the medium and μ_0, the permeability of free space.

When working in an environment where the magnetization can be disregarded, for instance, in vacuum, one has a straightforward linear relation between the magnetic induction and the magnetic field, Eq. 2.4.

$$\vec{B} = \mu_0 \vec{H} \tag{2.4}$$

2.1.3 Types of Magnetic Behaviours in Materials

Every material found in nature has some magnetic behaviour.

One crucial parameter when classifying materials magnetically is the susceptibility, χ - defined as the Eq. 2.5, χ is only a scalar if $\vec{M}$ and $\vec{H}$ are parallel; otherwise it will be a tensor.

$$\chi = \frac{\vec{M}}{\vec{H}} \tag{2.5}$$

As one can see by the mathematical definition, the susceptibility reflects how the material's magnetization reacts to an imposed magnetic field.

This section serves as a brief description of some of the most common magnetic materials found in MR devices. Ferrimagnetism is not addressed in this Thesis.

Please bear in mind that all materials discussed below have some sort of dependency with temperature; the discussion below is for a temperature inferior to Néel's temperature.

2.1.3.1 Ferromagnetism

In this class of materials, individual atoms have a net magnetic moment different from zero. In this case, there is a short-range, yet powerful interaction between neighbors atoms called exchange, this interaction, which has its origin in quantum mechanics, leads to a positive parallel alignment of neighboring magnetic moments. Ref. [15]

In this case, the magnetization describes a complex hysteresis curve. Fig. 2.2.

2.1.3.2 Antiferromagnetism

An antiferromagnetic material's atoms have a permanent magnetic moment such as the ferromagnetic case, but in this type of materials, there is an opposite relation between different magnetic moments when compared to the case of ferromagnetic materials. In this case, the neighbour magnetic moments orient in an antiparallel configuration. Ref. [15]

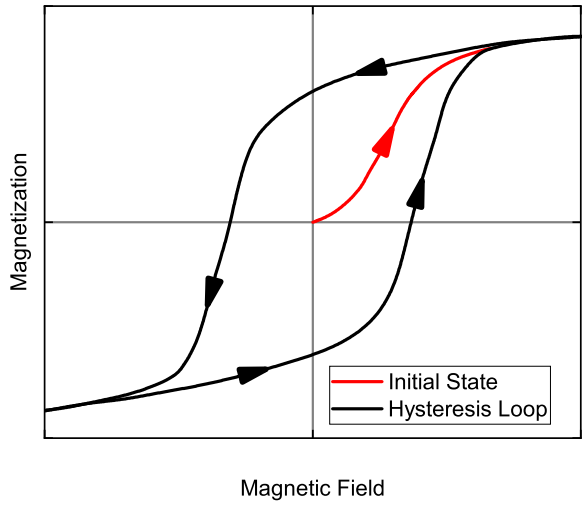


Figure 2.2: Magnetization *vs.* magnetic field for ferromagnetic materials. In red, one can see the initial state curve and, afterward, in black, the normal state hysteresis loop.

2.1.3.3 Diamagnetism

Diamagnetic materials exhibit the following behaviour; when one increases the magnetic field applied to them, their magnetization decreases linearly, Fig. 2.3. This class of materials is the one where the electronic shells are completely filled or the ones that have no effective magnetic moment. One very common material that is used in MR sensors as a spacer is copper; this is an example of a diamagnetic material. Ref. [15]

When reading about MR devices, it is common to find layers called non-magnetic. In reality, those layers are typically composed by diamagnetic materials.

Bear in mind that the relation is only linear if the temperature is kept constant along the process.

2.1.3.4 Paramagnetism

This class of material is similar to the diamagnetic class but in this case as one increases the magnetic field applied to them, their magnetization increases linearly.

In this case, the materials exhibit no net magnetization at zero field; however, if a magnetic field is applied, some of the magnetic moments align with the imposed field, thus increasing the magnetization. Fig. 2.3.

Similarly to what is stated for diamagnetic materials, the linear relation is only valid if the temperature is kept constant along the process. Ref. [15]

2.1.4 Artificial Neural Networks

Artificial neural networks (ANN) have been made famous because of their broad spectra of applications. Many challenges like image recognition, Ref. [19], pattern recognition, Ref. [20], time series

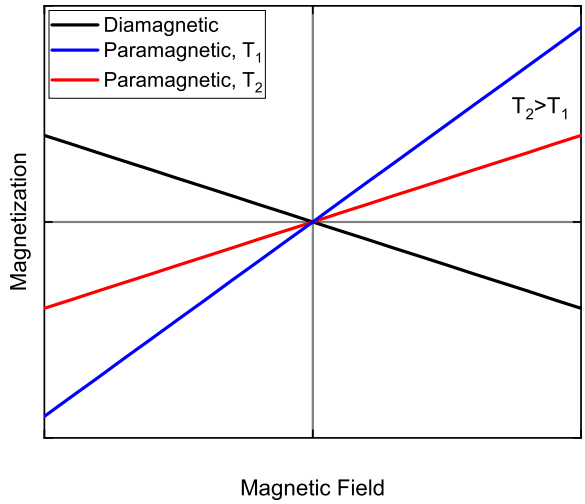


Figure 2.3: Magnetization *vs.* magnetic field for diamagnetic and paramagnetic materials. The effect of temperature is shown for a paramagnetic materials where $T_2 > T_1$.

forecasting, Ref. [21], medical diagnosis, Ref. [22], among others, have been found to have reliable solutions using ANN.

In December of 1943, Warren S. McCulloch and Walter Pitts published for the first time how neurons might work from the biological/mathematical point of view. They described some logic circuits that could mimic some behaviors of a network of neurons. Ref. [23]

Since then, the field has been evolving rapidly, and one has complex challenges with machine learning solutions. One type of network proven to be useful in several challenges is the Convolutional Neural Network (CNN); as the name states, this type of ANN includes one or more convolution layers.

Convolution, in the context of a neural network, is a layer that receives its input from more than one output of the preceding layer, meaning that they form a neighborhood in the previous layer. After having the neighborhood, a linear operation is done by multiplying the input values by a set of weights. The product is done by multiplying the input data, which is usually a two-dimensional array, with a two-dimensional array of weights called filter or kernel. Since one uses the dot product, the outcome of such an operation is just a value. The kernel is then swept across the input; Depending on the dimensions of the kernel and the input, the convolution's output can be an array of values. Fig. 2.4 illustrates an example of a convolution. Refs. [24, 25]

The convolution layer is of significant importance when one is trying to recognize a specific feature across one dataset. The convolution is invariant to translation; this becomes very handy if one is trying to detect specific features across multiple locations in a dataset. By applying the same kernel to different subsets of the dataset, one can identify whether the same feature is present or not. This important characteristic is essential in facial recognition, where one can use it to detect faces or specific elements of a face like a nose, for example. Refs. [24, 25]

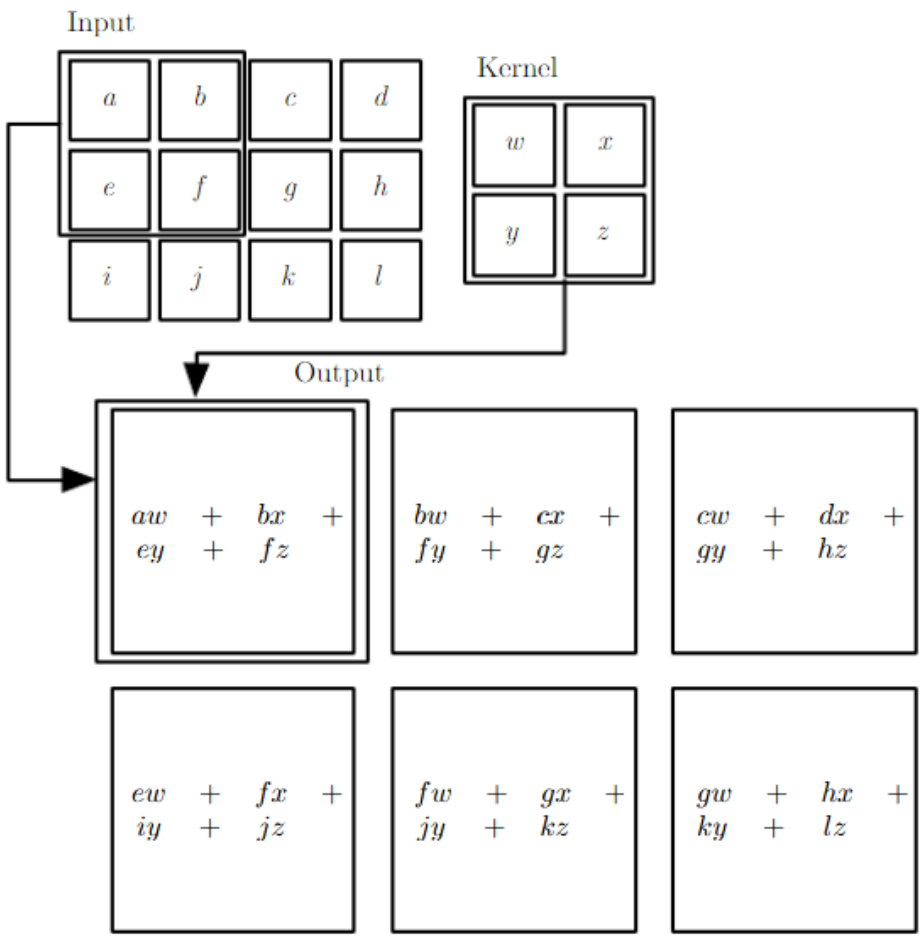


Figure 2.4: 2-D convolution example. One may see on the output the application of the kernel to every set of pixels in the image. Adapted from Ref. [24]. ©2016 MIT Press.

2.1.5 Transfer Curve

A transfer curve describes how the magnitude of a physical quantity is translated into an electrical signal by a given sensor. In a MR device this curve translates the sensed magnetic field into resistance values.

Fig. 2.5 shows what would be the transfer curve of an ideal sensor. As one can see, the perfect sensor should have a linear behavior between the two resistance plateaus, R_{High} and R_{Low}. For each point, ideally, there should be a unique correspondence between magnetic field and resistance.

The magnetic field, after which the plateaus are attained, is called saturation field (H_{sat}). Another important quantity when analyzing transfer curves is the magnetic coercivity (H_c) which is defined as the broadness of an hysteric curve.

In Fig. 2.5, there is a representation of what is happening between the angle of magnetization of the Free Layer (FL) with respect to the magnetization of the Reference Layer (RL). The antiparallel state is the one with high resistance, and the parallel configuration state is the low resistance one. In-between the two states, one has a linear region that can be used to sense magnetic fields.

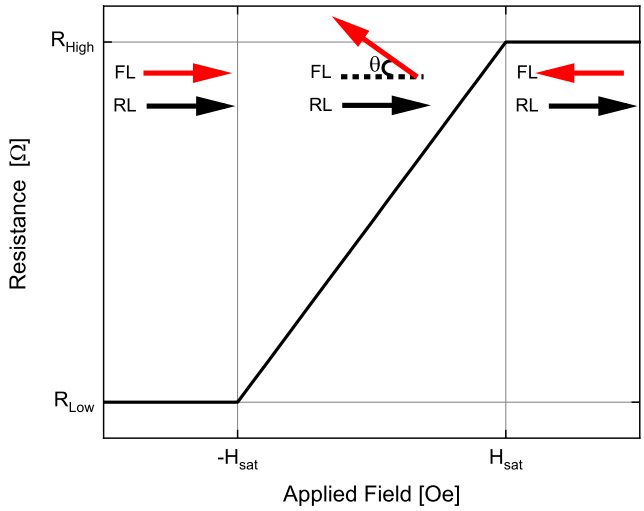
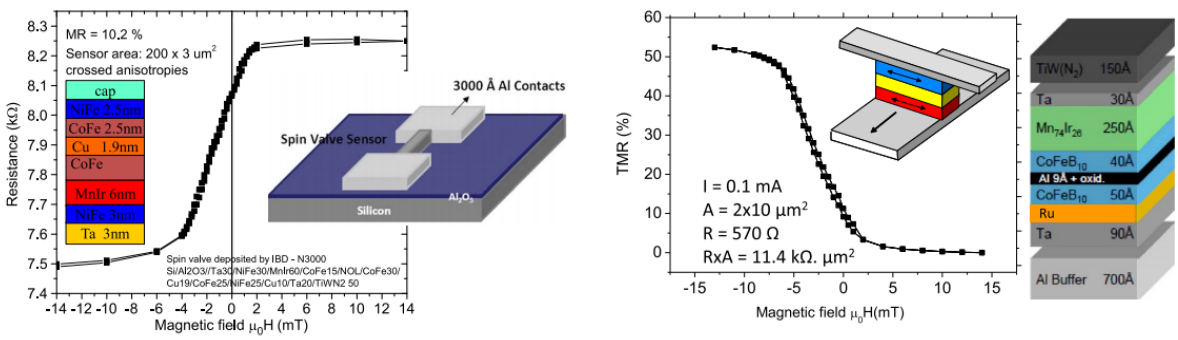


Figure 2.5: Transfer curve of an ideal sensor. The arrows represent the angle between the magnetization of the Free Layer (FL) and Reference Layer (RL).



(a) GMR based sensor, transfer curve, stack and architecture.

(b) TMR based sensor, transfer curve, stack and architecture.

Figure 2.6: Transfer curve for a GMR and TMR sensor. Stack composition also displayed. Adapted from Ref. [7]. ©2016 IEEE.

In Fig. 2.6, two different transfer curves are visible for a GMR and TMR based sensors alongside with the stack composition and architecture.

The curves can be mirrored with respect to the y-axis; The only difference is the direction of the applied magnetic field relative to the reference of the sample.

When fabricating MR sensors one aims to have all the sensors with a $R(H)$ loop similar to Fig. 2.5.

2.2 Magnetoresistive Sensors

The MR sensors produced at INESC MN use typically either a GMR or TMR configuration. The most common type of GMR based devices are Spinvalves (SV), and TMR based devices are called Magnetic Tunnel Junctions (MTJ). On Tab. 2.1, one may see a summary of the information on GMR/TMR sensors respecting physical principles behind different technologies, layer structures, and typical MR values.

	GMR (Spinvalves)	TMR (Magnetic Tunnel Junctions)
Physical principle	Spin-dependent scattering of conductions electrons	Spin-depedent tunneling of conduction electrons
Layer strcuture	Multilayer Reference/NM Metallic Spacer/Free layer	Multilayer Reference/Tunnel Barrier/Free layer
Magnetoresistance	$\approx 6 - 20\%$	$\approx 50 - 200\%$

Table 2.1: Typical properties of GMR and TMR sensors. Magnetoresistance range values taken from the Ref. [7]

When evaluating an MR sensor, different physical parameters are considered figure of merit, depending on the application. Some examples:

- MR - Magnetoresistance is defined as above, Eq. 2.1. The larger the MR, the larger the relative range of resistance that is available. Enabling easier signal interpretation by acquisition electronics;
- Linear Range - Range of field values where the sensor response varies linearly with the applied magnetic field. Ideally, for sensing applications, there should be a unique correspondence between the magnetic field sensed and resistance in this range of field values;
- Sensitivity (S) - It is commonly defined in the linear region as the quotient between the MR and the difference between the two saturation fields (H_{Sat}). It reflects the variation of MR with the magnetic field applied in the linear region. In Eq. 2.6 MR is the magnetoresistance, and ΔH is the linear region (ΔH will be equal to two times the saturation field, H_{Sat}).

$$S = \frac{MR}{\Delta H} \tag{2.6}$$

2.2.1 Fabricating a Magnetoresistive Sensor

Fabricating an MR-based sensor is a complex task. There are many possible manufacturing processes depending on the type of structure, the stack materials, dimensions of the features, etc.

This Thesis will focus on the process of an MTJ based sensor. Although considering an MTJ process, even for SV and other types of MR structures, there are manly three types of fabrication steps that one can use to produce these devices; these are: (i) The deposition of materials (films); (ii) Lithography; (iii) Etching. In this section, a brief explanation of each type of process and its characteristics are given.

Let us consider Fig. 2.6(b); by analyzing the stack, one sees the following structure (all thicknesses in $[nm]$):

70 Al / 9 Ta / Ru / 5 $CoFeB_{10}$ / 0.9 Al + oxi. / 4 $CoFeB_{10}$ / 25 $Mn_{74}Ir_{26}$ / 3 Ta / 15 TiW(N_2)

To build such a stack, one requires many different depositions of materials (films). Observing Fig. 2.6(b), one understands that patterning (by lithography) and removing materials (by etching) are required.

2.2.1.1 Thin Films Deposition

To manufacture devices, one has first to deposit the materials that will compose them. In particular, as these layers are typically very thin, ranging from just a few angstroms up to hundreds of nanometers, the process is typically referred to as the deposition of thin films. Refs. [26, 27]

If something wrong occurs during these steps, it can compromise critically the final device. Some of the typical challenges include incorrect deposition parameters, bad uniformity of the deposition, roughness of the deposited layer, etc.

One of the critical parameters that can be used to access the quality of such a process is the uniformity defined as:

$$uniformity = \frac{t_{\text{Maximum thickness deposited}} - t_{\text{Minimum thickness deposited}}}{2\left(t_{\text{Maximum thickness deposited}} + t_{\text{Minimum thickness deposited}}\right)} \times 100$$

Hence a smaller value of $uniformity$ means a more uniform process.

The techniques used to perform thin film deposition are usually separated into two different categories: Chemical Vapor Deposition (CVD) and Physical Vapor Deposition (PVD). In CVD techniques, reactant gases are introduced into the deposition chamber, and chemical reactions between the reactant gases on the substrate surface produce the film. The PVD is a family of methods where transfer of linear momentum by sputtering is used to produce the atoms which are deposited in the substrate. Refs. [26, 27]

One of the common ways to create the flux of atoms necessary through PVD is to through the ion bombardment of a plate of the material, typically referred to as the target. The flux of energetic ions that hits the target is typically created through a plasma. Refs. [26, 27]

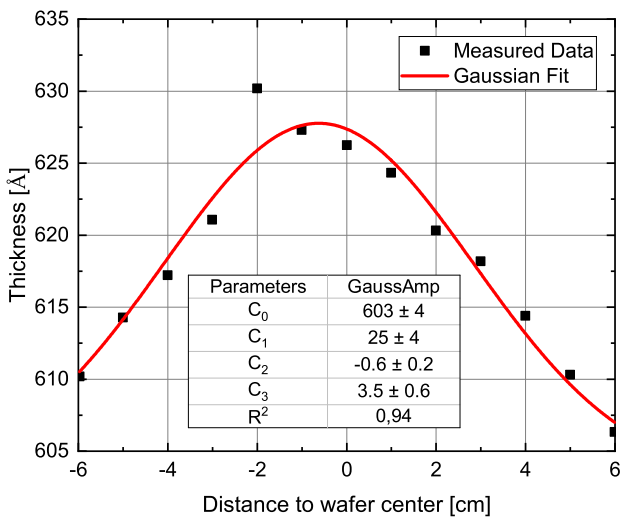
2.2.1.1.1 Ion Beam Sputtering Deposition

Ion Beam sputtering Deposition (IBD) is a method that belongs to the PVD family. The ion beam leaves the deposition gun (Kaufman Ion Source) aimed at the targets that are placed in a rotating prism mount. The material on the target in front of the deposition gun (unshielded) will be the material deposited on the sample. The ion beam hits the target with high energy. A part of its energy will be transferred to the target, and the other part dissipated as thermal energy. If the energy transferred to the target is larger than the material's binding energy, atoms will be released from the target. The target is at a specific angle to maximize the number of atoms that will reach the sample. When the atoms reach the sample's surface, placed in a rotating table, they will travel to places of minimal energy, staying there and forming a thin film. An assist gun could be also bombarding the sample while this deposition happens; this is used to increase the depositing species' kinetic energy, leading to a thin film coating with a modified density.

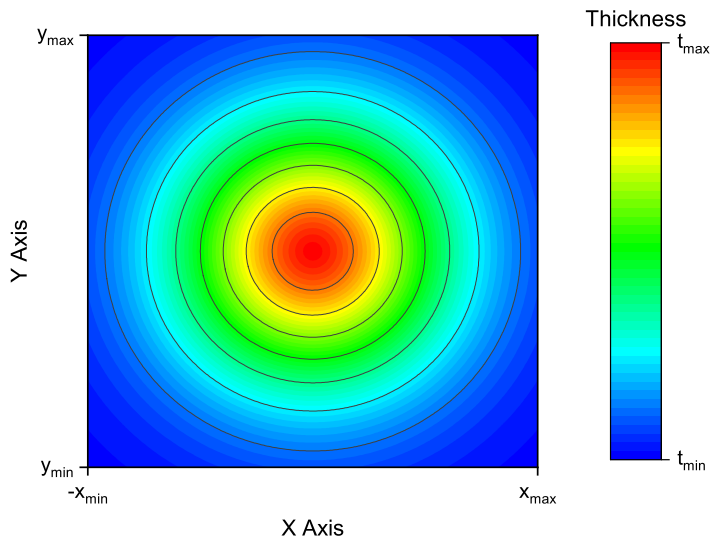
Due to the substrate table rotation, the thickness of the deposited layer will have radial symmetry; which can be modulated by a Gaussian function due to the radial nature of the process through which the beam of ions is created. A proof of the Gaussian nature of the beam can be seen in Ref. [28].

$$t(r) = C_0 + C_1 e^{-\frac{(r-C_2)^2}{2C_3^2}}, \quad C_0, C_1, C_2, C_3 \in \mathbb{R} \tag{2.7}$$

In Fig. 2.7(a), one can see the deposited thickness as a function of radius for the deposition of MgO using the Nordiko 3600, an IBD-based machine. Even though there is an outlier point, it is clear that there is a Gaussian-like behavior. Based on it a simulation of a contour plot (Fig. 2.7(b)) was performed, representative of what will be the radial symmetry of a deposited wafer; to drawn it, the constant C_2 was set to zero.



(a) Measured data of a deposition of MgO using Nordiko 3600 (IBD).



(b) Contour plot made with base on the fit from Fig. 2.7(a).

Figure 2.7: Deposition and Gaussian fit of measured thickness of a deposition of MgO in a 6" wafer at INESC MN using Nordiko 3600.

2.2.1.1.2 Direct Current/Radio Frequency Sputtering

DC (Direct Current) and RF (Radio Frequency) sputtering both work on basically the same physical principle. One starts by creating a plasma near the target that one wants to deposit on the sample.

To create this plasma, one starts by inserting a noble gas in a vacuum chamber, and with an RF or DC supply, one can excite the gas. By creating the plasma near the target, some atoms/molecules of the target will be removed and deposited on the sample. To conduct the target atoms/molecules to the sample, one connects the substrate table (where the wafer is placed) to ground.

When depositing non-conductive materials like oxides, one should use RF sputtering because DC would be ineffective in the sense that charges would get trapped in the surface of the oxide at the top of the sample, and the deposition rate would eventually drop to zero.

Due to the nature of the process, Eq. 2.7 can be used to express the deposited thickness (t) as a function of the radius (r).

2.2.1.1.3 Magnetron Sputtering

This is more efficient than the technique above; in this case, in order to focus the plasma in the zone of the targets, magnets are used to confine the plasma near that region.

Typically this technique is preferred to the simple RF/DC sputtering since this is more efficient since confining the plasma to the target zone leads to more interaction between the gas and the target.

Once more, the deposited thickness (t) as a function of the radius (r) can be described by Eq. 2.7.

2.2.1.2 Lithography

Lithography is the step where the wanted pattern is defined in a sample. The lithography needs three elements, a radiation source, a resist-coated sample, and a system to control how the sample is exposed to the radiation. Ref. [29]

Many techniques can be used for this step, which is crucial in terms of the lateral size definition of the structures to be patterned. Nowadays, there is an uprising tendency to use Extreme Ultraviolet Lithography (EUV) in the industry because it allows for patterning sub-10 nm structures. Ref. [30]

At INESC MN, the techniques available to do lithography are Direct Write Laser (DWL) Lithography, Electron Beam Lithography (EBL), and UV lithography. The latter uses hard masks. Both the EBL and DWL use virtual masks, allowing to design features with minimal size up to 50 nm and 1 μm respectively.

2.2.1.2.1 Direct Write Laser Lithography

Direct Write Laser Lithography is a commonly used technique for research proposes, because as patterns are being tested and optimized, there is no need to produce large volumes of samples, and thus not requiring a physical/hard mask is more adequate. In this technique, a laser is scanned across the sample covered in photoresist (PR). To successfully focus the laser on the sample covered with PR, a complex system of lenses is used. The PR is a compound that suffers a chemical reaction when exposed to the laser. There are two possible outcomes of the radiation interaction with the PR, the exposed

area either becomes *weaker* (the chemical connections are weakened) and therefore the exposed area becomes more soluble in the PR developer, or the exposed area becomes *stronger* (existing chemical connections are fortified) and hence the PR developer can dissolve the unexposed area of the PR. The PR is said to be of positive tone if it becomes soluble when exposed to light, and it is said to be of negative tone otherwise. Ref. [29]

A limiting factor of this setup and of optical lithography arises from diffraction limits and the optical system used. Recent efforts have been made to try and lower the wavelength of the radiation used which is 450 nm in the Heidelberg DWLii available at INESC MN.

Another challenge of this step might result from a non-uniform profile of the laser. That would cause the PR not to be fully affected by the radiation and hence not be correctly developed in the developing step. Another limiting factor of this step is the profile of the PR. While coating, different profiles of the PR might arise due to the amount of PR dispensed, erroneous spinner rotational speed, non-optimized baking, etc.

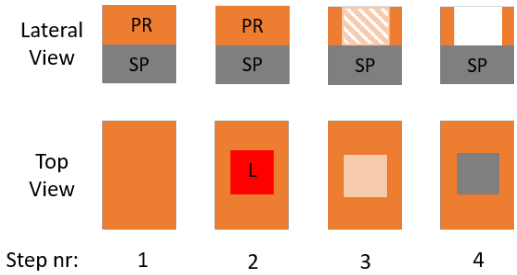


Figure 2.8: The cross-section information is available alongside with a top view. SP represents the sample, and L the laser.

In Fig. 2.8, is presented the lithography procedure. The first step is to prepare the sample by covering it with PR. The second step is where the pattern is passed onto the sample by a laser which is swept across the sample according to the virtual mask. In the third, one sees the sample after the writing procedure, where the PR in the exposed area is affected and thus then dissolved in fourth step, with PR developer (positive tone PR). If the PR was of negative tone then the unexposed area would be dissolved with the PR developer.

2.2.1.3 Etching

Etching is a process through which material is removed from the sample. There are two families of etching techniques, typically referred to as either dry or wet etching. In wet etching techniques, liquid-phase etchants are used to remove material, and in dry etching, a plasma or gas is used instead. Dry etching can be physical (through moment transfer) chemical or a combination of both. Refs. [10, 29]

2.2.1.3.1 Ion Beam Milling

In an Ion Beam Milling system, which is a technique of dry physical etching, a beam of ions, typically Ar^+, is used to bombard the sample, thereby transferring their momentum to the atoms on the sample, causing them to be removed from it. Refs. [10, 29]

In this Thesis, I will focus on Broad Ion Beam Milling in the sense that when milling, one is dealing with a broad beam of ions that remove the matter from a large area of the sample's surface. A review on Focused Ion Beam Milling may be consulted on Ref. [31].

During this procedure, the substrate table is rotating to increase the uniformity of the process. However, some challenges result from the Gaussian nature of the beam of etchant species. Another common challenge with this etching technique is the redeposition of material during the etching. Considering an MTJ, some metal may redeposit itself near the oxide barrier causing short circuits during the pillar etch. Other challenges of the Ion Beam Milling are the lateral profile of etched volumes and trenching, which will not be covered in this Thesis. Ref. [32] - Chapter 2.

There are many dependencies in the Ion Beam Milling etching process. If the sample is unpatterned, then one expects the uniformity across the etched surface to have radial symmetry. Once more, one can use Eq. 2.7 to express the profile of an unpatterned sample after etching if one discards the challenges listed above and consider only the Gaussian profile. Refs. [33, 34]

2.2.1.3.2 Reactive Ion Etching

This is similar to the system above meaning that it is a dry etching technique, but instead of just transferring momentum from the ions to the sample to release atoms/molecules, one also uses chemical reactions to facilitate their release. Refs. [10, 29]

Using chemicals can improve selectivity, etching rates, and also the uniformity of the process. The chemicals used should change according to the material that one wants to etch; for instance, for SiO_2 and Al, one can use C_4F_8 and $SiCl_4$ respectively. Refs. [35, 36]

There are two main challenges in dry chemical etching that can compromise the uniformity of the process. One is known as microloading, and the other is the Aspect Ratio Dependent Etch Rate (ARDE). Ref. [29]

(a) Microloading example. (b) ARDE example.

Figure 2.9: The two main challenges in dry etching that can compromise the process uniformity.

The microloading, Fig. 2.9(a), describes the etch rate's local dependence on the density of features patterned on the photoresist. A large unmasked surface area consumes more etching ions than a smaller unmasked area.

ARDE, Fig. 2.9(b), describes the etch rate's dependency on the ratio between the feature size and feature depth; the smaller the features become, the less probable it is for an ion to hit the feature and, therefore, the etching rate lowers.

As one can see, there are many dependencies in the Reactive Ion Etching process. One has the dependency with the features densities and distribution, turning the method of predicting the outcome of such a step a problematic task.

2.2.2 Summary of Possible Challenges and Metrology tools

Process	Possible Challenges	Metrology tools @ INESC MN
Deposition	Incorrect deposition parameters Incorrect profile of thickness or roughness Unsuited composition of the deposited layer	Profilometer Scanning electron microscopy Ellipsometer Optical microscope X-ray diffractometer Resistivity measurement
Lithography	Miss alignment between lithography levels Incorrect exposure to radiation	Profilometer Optical microscope
Etching	Incorrect thickness etched Incorrect profile of the etched layer	Profilometer Scanning electron microscopy Ellipsometer Optical microscope Resistivity measurement

Table 2.2: Summary of most common challenges that can happen in each of the processes.

As one can see by the Tab. 2.2 there are many features to control in a micro/nanofabrication process. This, alongside the several steps it takes to produce a given MR sensor, makes predicting the yield of a micro/nanofabrication a rather challenging task.

Chapter 3

Treating and Classifying Autoprober Data

This chapter discusses a novel and faster way of interpreting the data retrieved from measurements performed in the Autoprober ($R(H)$) installed at INESC MN. A Graphical User Interface for data retrieveal and data vizualization, that the user can customize was developed. Additionally, it is also presented a transfer curve classifier for sensing applications that can distinguish between four different classes of sensor quality.

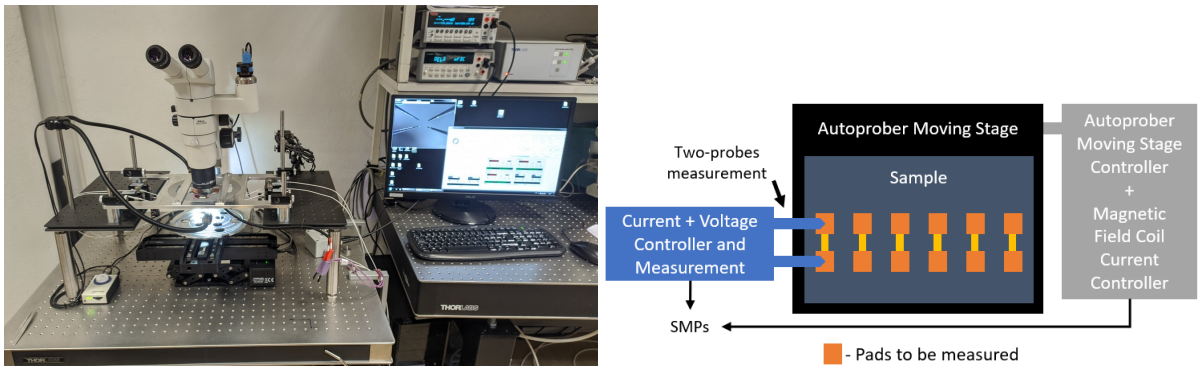
3.1 Autoprober Characterization

The Autoprober is a device that exists at INESC MN and consists of a stage that can move in three distinct directions, x, y, and z, allowing for automatic electrical characterization of devices while a magnetic field is imposed. To make the measurements, a map must be defined by the user containing the position onto which the measurement probes should be lowered. The setup allows a customized set of voltages/electrical currents and measures, respectively, electrical currents or voltages. The number of probes can be customized to 2 or 4, and the distance between tips can be tailored to each sample/wafer's needs. A dedicated 4 probe head, with fixed distance probes, allows mapping the electrical resistivity across a full wafer.

For each measurement the Autoprober stage is raised for the samples to contact the probes at a given map location, and then a set of $R(H)$ measurements are done, and the same is then repeated for all locations of the map. To impose the magnetic field, a set of coils meticulously calibrated exist near the position which is being measured. To impose a magnetic field, a current is sent through the coils; the maximum value of the magnetic field is $\approx$ 300 Oe.

The setup is set to measure up to 8" wafers by having a stage that can travel up to 220 mm in the x and in the y direction. The device returns a file of extension".SMP" of semicolons separated values. The file contains many columns, and each row corresponds to a given sensor, a given field, and by computing the quotient between voltage and electrical current, a given measurement of resistance.

To impose the current through the calibrated coils to generate the magnetic field, a Kepco BOP 50-4D power supply unit is used. As a multimeter to measure the real voltage and electrical current through the probes, an Agilent 34401A digital multimeter is used; to impose either a voltage or current through the probes, a Keithley 2401 source meter unit is used. The specification of each device can be consulted on their datasheets.



(a) Autoprober Setup. (b) Autoprober Setup Schematic.

Figure 3.1: Autoprober Setup and Scheme at INESC MN

Input 1	Input 1 Min	...	M8 Curr [A]	G Output [Oe]
PS3 DC Current (A)	-0.2:-0.05:0.05	...	-	-
PS3 DC Current (A)	-0.2:-0.05:0.05	...	-	-
...	...	...	...	...

Table 3.1: Example of a SMP file that results from the use of the Autoprober. For an easier reading, the data was formatted from a ".SMP" to a table, each semicolon represent a new column and only four columns of a total of thirty nine columns are shown. In order to keep the table small only two measurements of different field current applied are shown. This SMP has almost eighteen thousand rows (17852), only two are shown.

3.2 Autoprober Data Processor, Sensor Classifier & Visualizer

The first challenge tackled by this Thesis is developing a tool to handle the data outputted by the Autoprober at INESC MN. This tool should be simple, quick to use and user-friendly.

Another requirement imposed by users at INESC MN is that the code should run on any operative system, and hence the decision to address this problem with Python 3.8, Ref. [37]. I installed Anaconda, Ref. [38], because it includes most of the needed libraries for data science. On top of that I installed Keras/Tensorflow, Ref. [39, 40], and Plotly, Ref. [41]. I used Jupyter Notebook as mine integrated development environment (IDE), one of the most used in data science, Ref. [42]. Keras/Tensorflow was installed because of the curve classifier, Sec. 3.2.2 and Plotly was installed to make the interactive plots that are saved in .HTML format.

The code includes three distinct functionalities: (i) Data Processor; (ii) Sensor Classifier; (iii) Data Visualizer.

The Data Processor allows to process the ".SMP" files and retrieve relevant quantities (eg. MR) of MR devices. The Sensor Classifier uses a machine learning approach that can access large volumes

of data regarding $R(H)$ measurements of sensors and classify the loops between four different classes. Finally the Data Visualizer facilitates the analysis of the data returned by the two previous functionalities by returning a two-dimensional scatter interactive plot with the relevant information displayed.

Due to large volume of code[1] needed to build such a program, I could not include it in this Thesis; however, the complete code is available in the following reference (drive), Ref. [43].

In Sec. 3.4, one may see the installation guide to install all required packages to use the code.

3.2.1 Data Processor

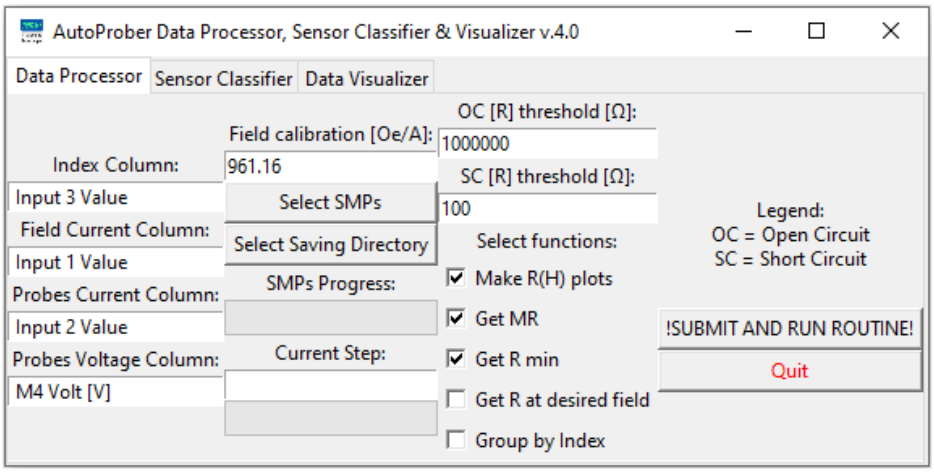


Figure 3.2: Autoprober Data Processor version 4.0 with the default values.

For the code to run, the user has to input the following data (please check Fig. 3.2 as some predefined values are shown there):

- **"Index Column"** - The column header that contains the index of measurement; this index is constant across the measurements of one sensor. This index is also called Map Index;

- **"Field Current Column"** - To generate a magnetic field, coils exist in the stage where the wafer is placed. This is the column with the values of current used for imposing a magnetic field using the coils. To obtain the field from the current that flows through the coils, one has to multiply the field current by the field calibration constant ($H = I$.constant);

- **"Probes Current Column"** - This is the column with the measurements of current flowing through the probes;

- **"Probes Voltage Column"** - This is the column with the measured voltage across the probes;

- **"Field calibration"** - At the date of release of version 4.0, this was the value of the calibration constant. To obtain the field from the current that flows through the coils, one has to multiply the current by this constant ($H = I$.constant);

[1] The code has nearly one thousand five hundred lines, it would fill at least thirty pages of this document.

- **"Select SMPs"** - Opens a pop-up windows for the selection of the SMP files. Many SMP files can be selected at the same time;

- **"Select Saving Directory"** - Opens a pop-up window to select a folder to save the computations processed by the code;

- **"OC [R] threshold:"** - Open circuit resistance threshold, if the average resistance of a measured sensor is larger than this threshold, then it is classified as an open circuit. This "filter" is only available on some functions. More information available on the **Functions Description** subsection below. The user can deactivate such the filter by inserting a high threshold value (e.g. 10M Ω);

- **"SC [R] threshold:"** - Short circuit resistance threshold, if the average resistance of a measured sensor is smaller than this threshold, then it is classified as a short circuit. This "filter" is only available on some functions. More information on the **Functions Description** subsection below. The user can deactivate such the filter by inserting a low threshold value (e.g. 1 Ω);

3.2.1.1 Functions Description

The code was built considering a modular approach so that upcoming users may adapt the code to their needs and easily create new functions.

The current working functions include:

- **"Make R(H) plots"** - This function draws $R(H)$ plots under a folder with path **"Saving Directory"/R_H_curves/SMP/SMP Name**. Inside this path, there is another folder named "**Data**" which contains all the ".csv" files which originated each plot. The ".csv" contains multiple columns and rows; the columns correspond to the current used to generate the magnetic field, the magnetic field, the resistance measured, and the bias current;

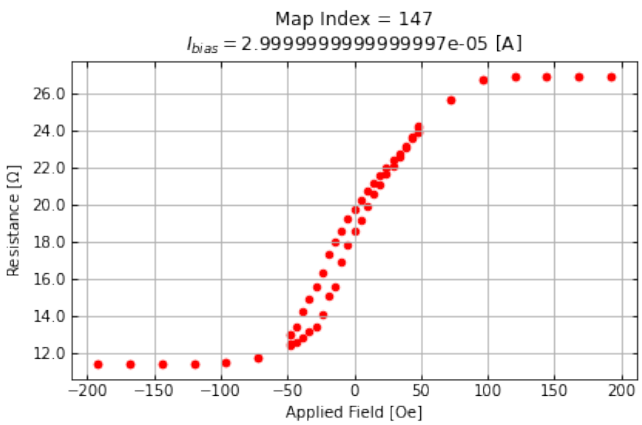


Figure 3.3: Example of an randomly selected $R(H)$ curve retrieved by the Data Processor.

- **"Get MR"** - As stated in the Sec. 2.2 one crucial parameter when classifying sensors is magnetoresistance. This function returns a ".xlsx" file with two columns, one corresponding to the Map Index and another corresponding to the computed magnetoresistance. The file is stored inside the

Field Current [A]	Applied Field [Oe]	Resistance [Ω]	Bias Current [A]
-0.2	-192.232	325.78126666666674	2.9999999999999997e-05
-0.175	-168.20299999999997	325.9209666666667	2.9999999999999997e-05
-0.15	-144.17399999999998	326.1114	2.9999999999999997e-05
...	...	...	...
-0.15	-144.17399999999998	326.1273666666667	2.9999999999999997e-05
-0.175	-168.20299999999997	325.9213333333334	2.9999999999999997e-05
-0.2	-192.232	325.7621333333334	2.9999999999999997e-05

Table 3.2: Example of a .csv file computed by the **Make R(H) plots** for a real device measured in the Autoprober. For an easier reading the data was formatted from a ".csv" to a table, each comma represent a new column. In order to keep the table small only six measurements are shown.

path **"Saving Directory"/MR/SMP Name**. For better statistics, the R_{High} (R_{Low}) is an average of the three highest (lowest) resistance values. This function includes an open circuit (OC))/short circuit (SC) filtering meaning that curves where the average resistance of the measured device is larger then the OC threshold provided by the user the MR instead of a value is the string "OC" and if the average resistance is lower than the SC threshold then it is classified as the string "SC". The number of resistance values to be averaged can be manipulated through the *n_to_average* variable at the code level;

Map Index	MR [%]
0	SC
1	117.8053413
2	95.05856497
...	...

Table 3.3: Small portion of an example excel file generated by the **Get MR** function.

- **"Get R min"** - Another essential parameter when qualifying MR sensors is the $R \times A$ product. This function generates a ".xlsx" file with two columns, one corresponding to the Map Index and another corresponding to the lowest resistance value (R_{min}). The ".xlsx" is saved inside the path **"Saving Directory"/Rmin/SMP Name**. Like for the previous functions, R_{min} is an average of the three smallest resistance values and the variable *n_to_average* can be changed. Again OC/SC filtering is included;

Map Index	Rmin [Ω]
0	SC
1	378.6732
2	325.8215
...	...

Table 3.4: Small portion of an example excel file generated by the **Get R min** function.

- **"Get R at desired field"** - Some applications may require specific values of resistance at certain fields. This function returns two different ".xlsx" files; one terminates with "_avgs.xlsx" and another with "_both.xlsx". The functions opens a pop-up window, which asks the user to input the field at which to get the resistance. The function will find the two magnetic field values closest to the requested one and return the corresponding values of resistance. File "(...)_both.xlsx" stores both

values pairs whereas the "(...)_avgs.xlsx" return the average of both the field and the resistance. The "(...)_avgs.xlsx" is advantageous for the Data Visualizer and includes OC/SC filtering;

Map Index	Applied Field [Oe]	Resistance [Ω]
0	0	SC
1	0	516.146
2	0	437.1613
...	...	...

Table 3.5: Small portion of an example excel file generated by the **Get R at desired field** function, in this case the goal field was one oersted, the closest real value measured was zero oersted. In this case the file is the one with the average values "(...)_avgs.xlsx".

Map Index	Applied Field [Oe]	Resistance [Ω]
0	0	25.33353
0	0	25.38567
1	0	422.4394
1	0	609.8527
2	0	387.6377
2	0	486.6848
...	...	...

Table 3.6: Small portion of an example excel file generated by the **Get R at desired field** function, in this case the goal field was one oersted, the closest real value measured was zero oersted. In this case the file is the one with both values "(...)_both.xlsx".

- **"Group by Index"** - Sometimes, some specific applications may require more than one measurement per die/chip. When using the Autoprober one way in which one can optimize time-consumption is to measure the same set of pads across all dies/chips, this is optimal because the same set of pads will have the same interdistance in all equal dies/chips. Considering N different dies/chips that need n measurements taken per dies/chip, meaning that one has n SMP files each with N measurements; after taking the measurements one might want to analyse the $R(H)$ loops by die/chip and not by the SMP order, this functions allows to organize the $R(H)$ loops by index of measurement (Map Index). The organized folders are saved inside the directory **"Saving Directory"/R_H_curves/Index**/.

3.2.2 Curve Classifier - Machine Learning Approach

My Master Thesis's second goal was to build a tool to facilitate the $R(H)$ characterization of large volumes of data for sensing applications. I developed a neural network to do an automatic classification of the curve shape which is faster and cheaper then performing it manually.

The goal of this classifier is to sort $R(H)$ curves of both GMR and TMR based devices into four different categories. Before being able to define the difference between the four different classes it is mandatory to talk about the data preparation.

It is crucial to understand that this classification only takes into account the shape of the curves and not the values of resistance/magnetic field. Therefore this should only be interpreted as complementary information.

3.2.2.1 Data Preparation

Before advancing to the model itself it is important to state what kind of treatment I gave to the data before trying to train such network.

To be independent of the stack deposited, there is the need to normalize both the resistance and the applied field between two values MAX and MIN. For this, I defined a function that implements the following mathematical operation to a list of n values $\{x\} = \{x_0, x_1, ..., x_n\}$:

$$\{X\} = (MAX - MIN) . \frac{(\{x\} - max\,(\{x\}))}{max(\{x\}) - min(\{x\})} + MAX$$

$\{X\}$ is the normalized list of $\{x\}$ between any value MAX and MIN. For this work I defined $MAX = 1$ and $MIN = -1$; therefore all the data processed by the code is now between -1 and 1 in both the x and y-axis. By doing this, one will have curves that are always majored between the same values, and hence, using this operation, one is increasing the amount of data for each class without losing the curve tendency. If one did not do this operation, one would need a much vast database to train the network since examples for each stack/each value of $R \times A$, etc, would be needed, and this would be unfeasible in the lifetime of my Thesis.

Other critical functions that I defined were an upsampling and downsampling functions. When acquiring data in the Autoprober different sampling points might be chosen. One characteristic of an ANN/CNN is that shape of the input layer cannot change from time to time; therefore, it is crucial to have upsampling and downsampling functions to downsample or upsample data to meet the requirements of the input layer.

In the dataset that I gathered, only the downsampling function was used. Almost all my data had 70 sampling points per curve. In the original data, 1020 curves have 70 sampling points, and 94 have 86 sampling points.

		# Occurrences
Category	**NOK**	618 (55.48%)
	A	325 (29.17%)
	M	73 (6.55%)
	OK	98 (8.80%)
	Sum	1 114 (100.0%)

Table 3.7: Table of occurrences of each class in the dataset.

In order for all models presented in the next section to be tested under the same conditions, the dataset for training, validation, and test were kept constant. Regarding the ratios used for each dataset, I used 60% of all available data for the training set, 20% for validation, and the remaining 20% for testing. Each of the datasets were picked randomly out of the 1 114 examples classified; the data was picked randomly but in a manner that each class has identical distributions represented in each of the datasets. This is crucial because the dataset used is imbalanced, as shown on Tab. 3.7.

One-hot encoding of the target was performed since there is no ordinal relationship exist between the different classes. One-hot encoding of a feature consists in the creation of new dummy features for each

		Category			
		NOK	**A**	**M**	**OK**
Dataset	**Train** (59.96%)	370 (55.39%)	195 (29.19%)	59 (6.59%)	44 (8.83%)
	Validation (20.02%)	124 (55.61%)	65 (29.15%)	19 (6.73%)	15 (8.52%)
	Test (20.02%)	124 (55.61%)	65 (29.15%)	20 (6.28%)	14 (8.97%)

Table 3.8: Category distribution across the three different datasets.

unique values of the feature column meaning that a target with n classes would lead to n new columns. Each of the variables are replaced by a new binary variable for each of the unique feature/target. By doing this, the model understands that the classes are entirely different categories. Ref. [25]

3.2.2.2 Classes

After treating the data I sorted all $R(H)$ loops into four distinct categories.

- **NOK** - This label is for curves where there is no visible sign of a MR behavior of the material. No visible plateaus and no visible linear response typical of a sensor;

- **A** - This is for curves where there is some MR behavior of the material but not as expected. As I was interested in sensors, the essential criteria that I used to include curves in this category were:

  – Large and irregular coercivity along the curve ($\gtrapprox 10\ Oe$) - Non-normalized plot;

  – Large Barkhausen jumps in the transfer curve ($\gtrapprox 0.25\ AU$) - Normalized plot.

- **M** - This is for curves where there is a MR behavior of the material but not of an ideal sensor. Criteria to include curves in this category was:

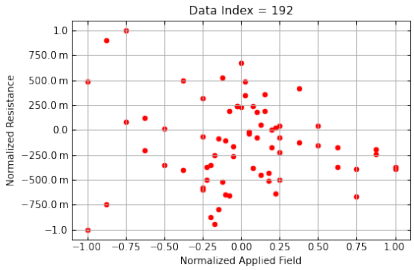
  – Small coercivity along the curve ($\lessapprox 10\ Oe$) - Non-normalized plot;

  – No large Barkhausen jumps in the hysteresis curve ($\lessapprox 0.25\ AU$) - Normalized plot.

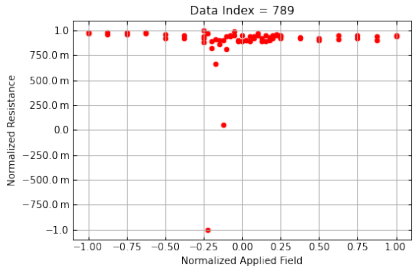
- **OK** - This is for curves where there is a MR behavior of the material typical of an ideal sensor:

  – No visible coercivity ($\lessapprox 1\ Oe$) - Non-normalized plot;

  – No visible Barkhausen jumps ($\lessapprox 0.04\ AU$) - Normalized plot.

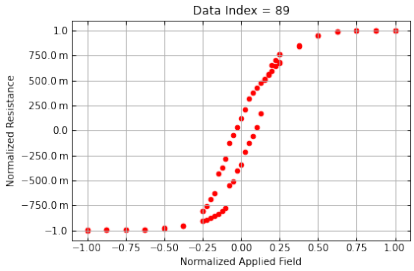


(a) Data Index 192.

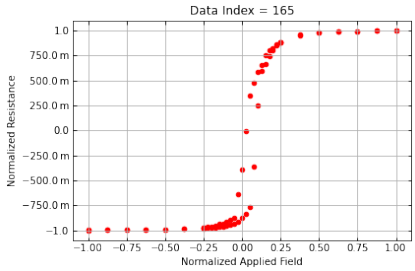


(b) Data Index 789.

Figure 3.4: Normalized **NOK** examples.

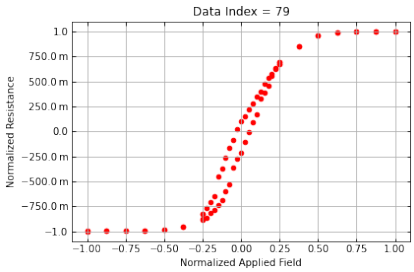


(a) Data Index 89.

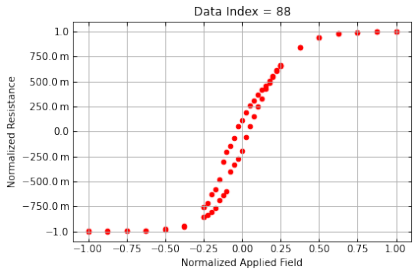


(b) Data Index 165.

Figure 3.5: Normalized **A** examples.

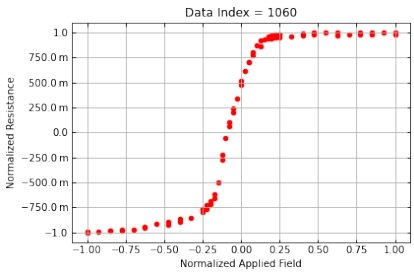


(a) Data Index 79.

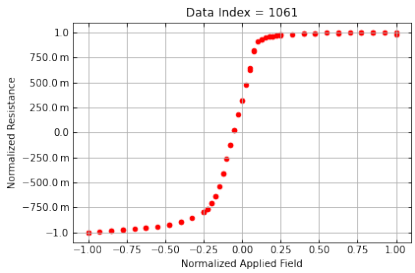


(b) Data Index 88.

Figure 3.6: Normalized **M** examples.



(a) Data Index 1060.



(b) Data Index 1061.

Figure 3.7: Normalized **OK** examples.

3.2.2.3 Model and Training

Regarding the models used, several architectures were tested, and the summary of what was tested is shown below. I tried 8 different models; to build them, I inspired myself in the MNIST problem. The MNIST problem consists of handwritten digit recognition and, as the challenge tackled by this Thesis is also pattern recognition, the solution to the MNIST problem provides pertinent results for this problem. Ref. [25, 44]

I constructed seven different CNNs and one ANN without any convolutional layer; on the Tab. 3.9 one may see the results across all models. The models layers can be seen in the Figs. 3.8 and 3.9.

Layer description:

- **Conv2d** - This is the convolution layer. This layer creates a convolution kernel that is convolved with the layer input yielding a tensor of outputs. Ref. [24] - Chapter 9, Ref. [25] - Chapter 15;

- **MaxPool2d** - This layer returns the maximum value within a mask that is defined by the user and then swept across the input layer. Ref. [24] - Chapter 9, Ref. [25] - Chapter 15;

- **Dropout** - The dropout layer randomly sets input units to zero; this helps to prevent overfitting while training. The amount of units to be set to zero is defined by the user. Ref. [24] - Chapter 7, Ref. [25] - Chapter 15.

I included an adapted version of the code used to train the models and test them, Appendix A.1. If needed the Ref. [39] - Keras layers Application Programming Interface (API) should be consulted for insights on the layers used.

Before being able to train any model a loss function should be chosen. The loss function is used to train the models by rewarding the correctly identified data and penalizing the missclassified cases in each epoch of the training.

After defining the loss, one optimizes each parameter by taking a small step in the opposite direction shown by each parameter's gradient concerning the loss. In reality, I used the Adam Optimization Algorithm, a commonly used optimizer for such application because it achieves good results faster than the classical stochastic gradient descent-based algorithms. I did not change any of the algorithm's parameters; all had their default values. Ref. [45]

I opted for the Categorical Cross-Entropy (CCE) as the loss function, defined as:

$$CCE = -\sum_{i=1}^{output} y_i.log\left(\hat{y}_i\right) \tag{3.1}$$

Where y_i in the real label of the $i-th$ index of the target array and $\hat{y}_i$ is the model prediction (probability). The minus sign guarantees that CCE gets smaller when y and $\hat{y}$ get similar, and that is exactly what a loss function should do.

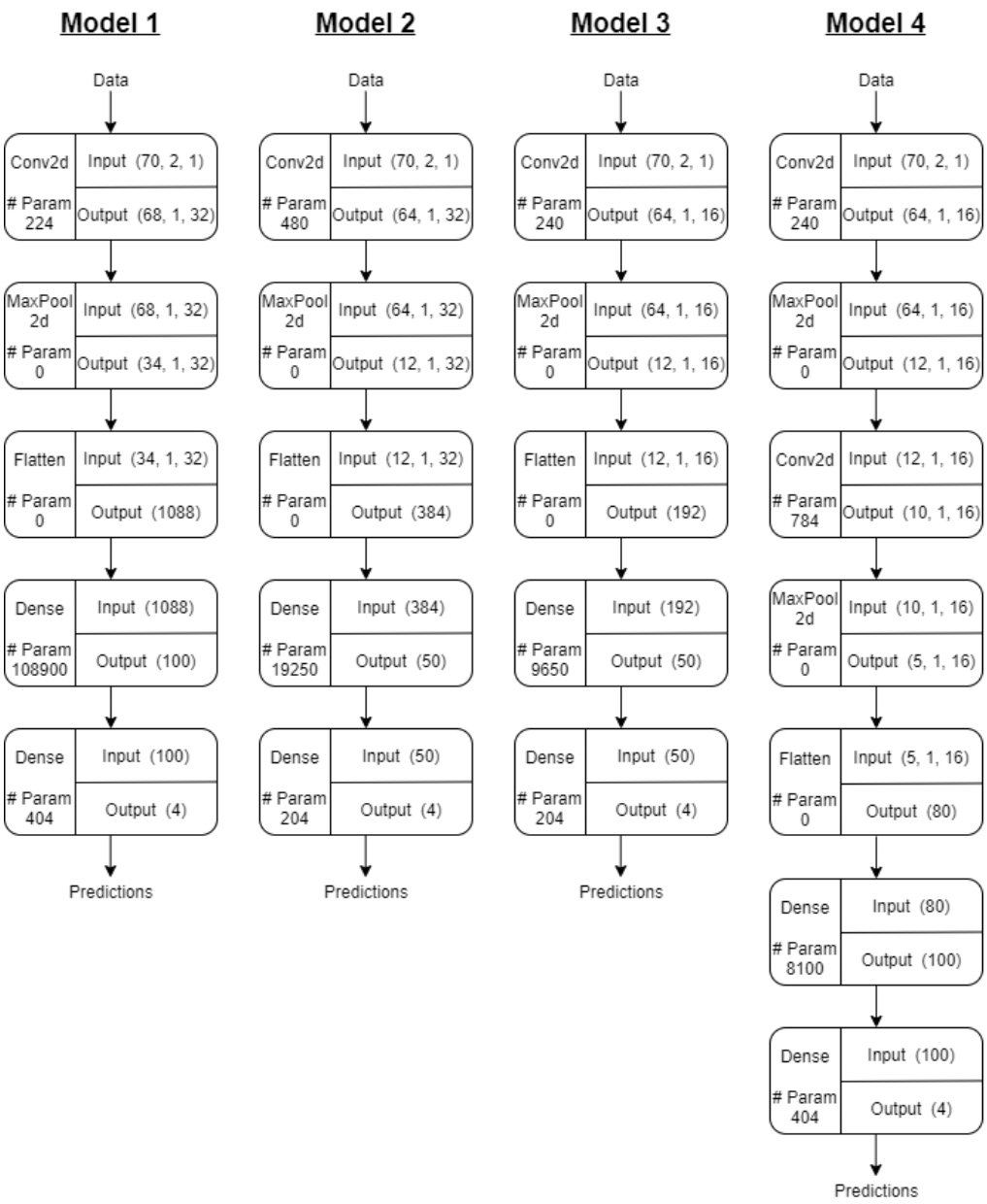


Figure 3.8: Layers of models 1-4. Details like kernel sizes, filters, etc., can be seen in the Appendix A.1

To prevent overfitting, I used early stopping. One problem when training ANN is choosing the number of epochs to train the model. Training the model too many epochs will result in an overfitted model that will predict the training dataset's noise and not the general tendency as one wishes. Few epochs will result in an underfitted model that will have no use to predict the wanted behavior. Early stopping allows getting the optimal number of epochs needed to train a model; for this, a validation dataset is needed. Early stopping checks the performance of the model in the validation dataset, once the performance of the model on the validation dataset stops improving, the training is stopped, and the best model is saved. To monitor the performance, the loss is calculated on the validation dataset. In this case, I set 50 to be the number of epochs, after which the training is stopped if no improvement occurs.

See Fig. 3.10 and check one example of training with early stopping and one where no early stopping is implemented.

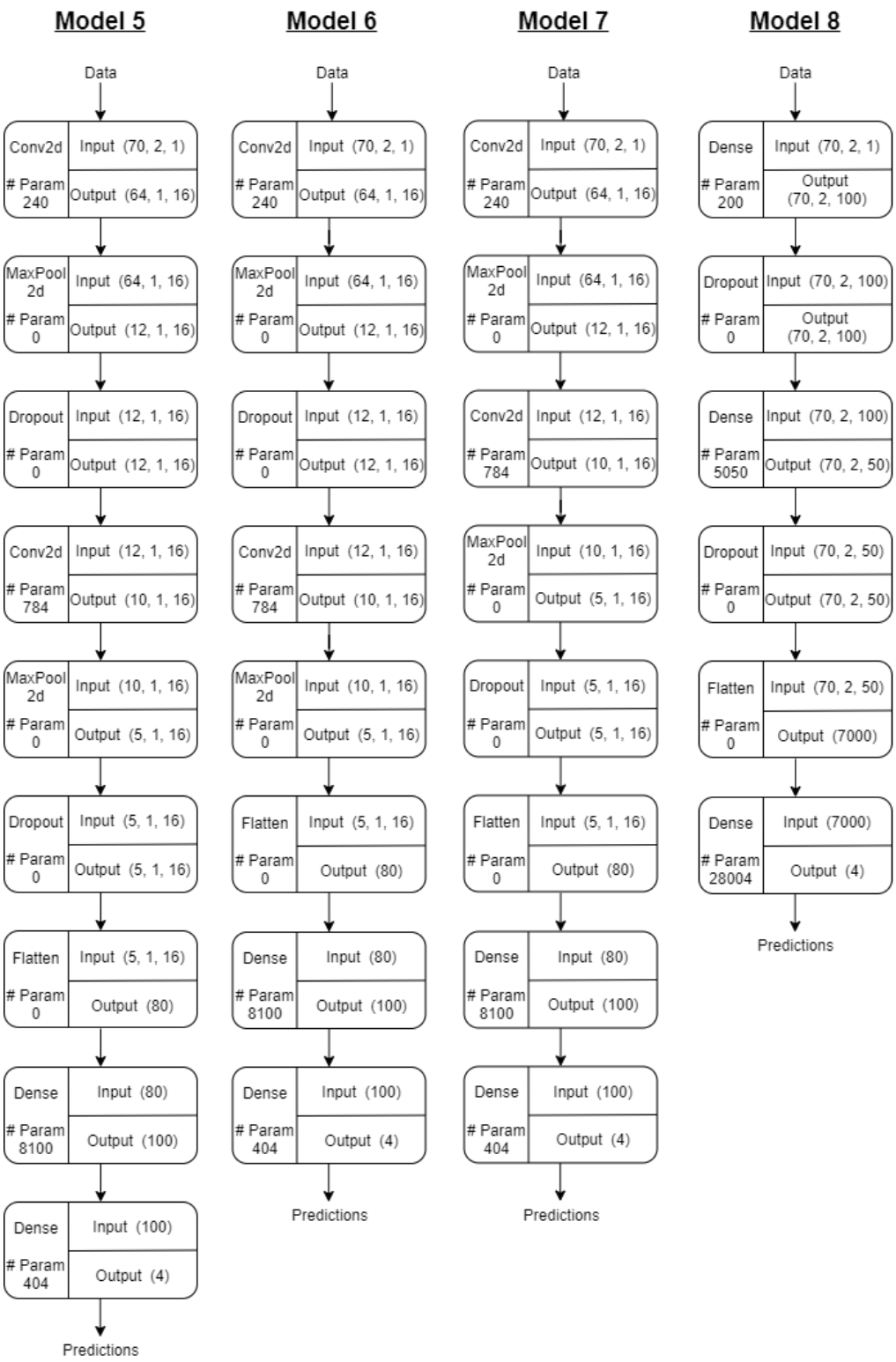
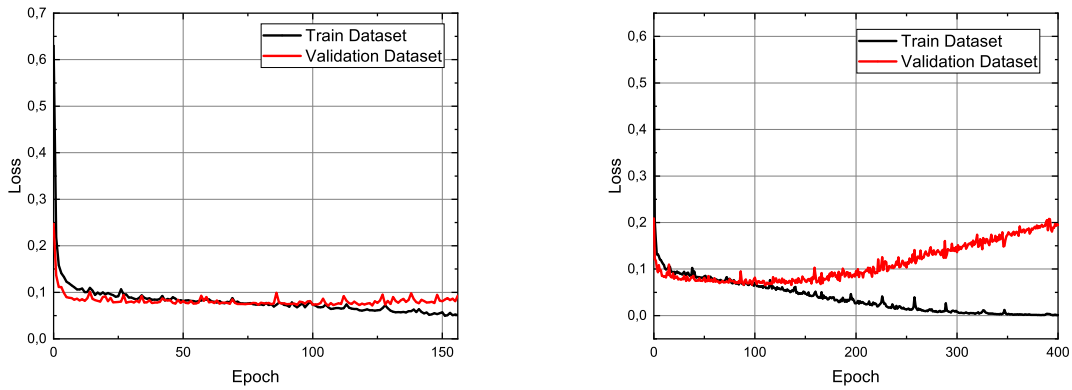


Figure 3.9: Layers of models 5-8. Details like kernel sizes, filters, etc., can be seen in the Appendix A.1



(a) Early stopping.

(b) No early stopping.

Figure 3.10: Early stopping *vs.* no early stopping example for model 1.

In Fig. 3.10(a), the curves do not have to be precisely equal until the final epoch since there is always a stochastic term due to the stochastic nature of the process of training a neural network.

3.2.2.4 Evaluation

After training the model, a vital step is to evaluate its performance; this should always be done in data that the model has not seen through the previous training process; this is why the test dataset is essential. When evaluating the models trained, I used two different metrics to assess the performance, the accuracy, and the F_1 score.

The accuracy is unarguably one of the most used metrics. It is mostly used when all the classes are equally important and when one has a balanced dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

Where TP are the true positives, TN are the true negative, FP are the false positives, and FN the false negatives.

However, the accuracy can be misleading when one has an imbalanced dataset. When evaluating an imbalanced dataset, and when the incorrectly classified cases are crucial one should look at other metrics such as the F1 score, which gives a better measure of the incorrectly classified cases than the accuracy. Ref. [25] - Chapter 6.

Accuracy should be used when the True Positives and True negatives are crucial while F_1 score is used when the False Negatives and False Positives are crucial which was the case. I did not want curves of defective sensors being classified as functional sensors (FP) and curves of functional sensors being classified as defective sensors (FN), for that reason I used the F_1 score.

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

$$F_1 = 2.\frac{Precision.Recall}{Precision + Recall} \tag{3.5}$$

To demonstrate the importance of the F_1 score take the classic binary cancer detection example. Ref. [25] - Chapter 6. If one has a dataset where a cancer only appears in twice out of one hundred times, and if the model classifies: (i) 98 patients that are healthy as healthy ($TN = 98$); (ii) 1 patient that is sick as sick ($TP = 1$); (iii) 1 patient that is sick as healthy ($FN = 1$). Then the model will have an accuracy of:

$$Accuracy_{\text{Cancer example}} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1 + 98}{1 + 98 + 0 + 1} = 0.99$$

However, the F_1 score would be:

$$F_1 = 2.\frac{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} = 2.\frac{\frac{1}{1+0} \cdot \frac{1}{1+1}}{\frac{1}{1+0} + \frac{1}{1+1}} = 0.67$$

Therefore, it is evident that in imbalanced datasets, and when one cares about the FP and FN one should also consider the F_1 score as an evaluation metric.

As the problem that I was dealing with is multiclasses, there are four different classes of targets, some modifications to the F_1 formalism are needed. I used the F_1 defined as the average of the computed F_1 score for each class without any weights to account for the imbalanced dataset. This might lead to a larger penalization if the classifier does not perform well with the minority classes. I choose such a definition because I was dealing with an imbalanced dataset.

	Accuracy	**F1 Score**	**Optimal number of epochs**
Model 1	0.9821	0.9566	8
Model 2	0.9821	0.9566	12
Model 3	0.9776	0.9442	7
Model 4	0.9865	0.9623	229
Model 5	0.9776	0.9442	80
Model 6	0.9776	0.9442	10
Model 7	0.9821	0.9540	219
Model 8	0.9686	0.9168	124

Table 3.9: Short evaluation table of models tested.

The Tab. 3.9 reveals that model 4 has the highest evaluation scores. This will probably lead to a better classifier. With this being said, I now have the architecture that I will use for the final model defined.

3.2.2.5 Final Model

Even though the optimal number of epochs may change from run to run due to the process's stochastic nature, I assumed that it would be the same for a not so different training dataset. Since I wanted to have a more reliable testing of my model, I trained once more a model with the same architecture as model 4, but now the training dataset can have 70% of all data, and the rest 30% can be saved for testing. This will lead to more reliable testing; however, I might not have the training stopped in the optimal epoch since I have no validation dataset for the early stooping.

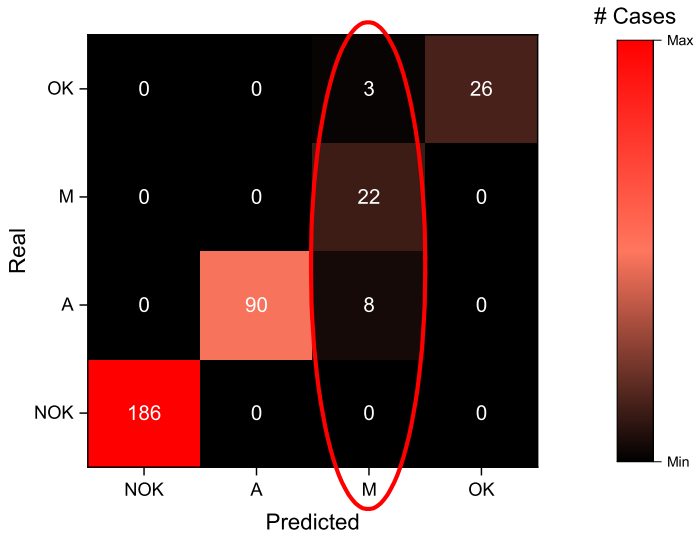
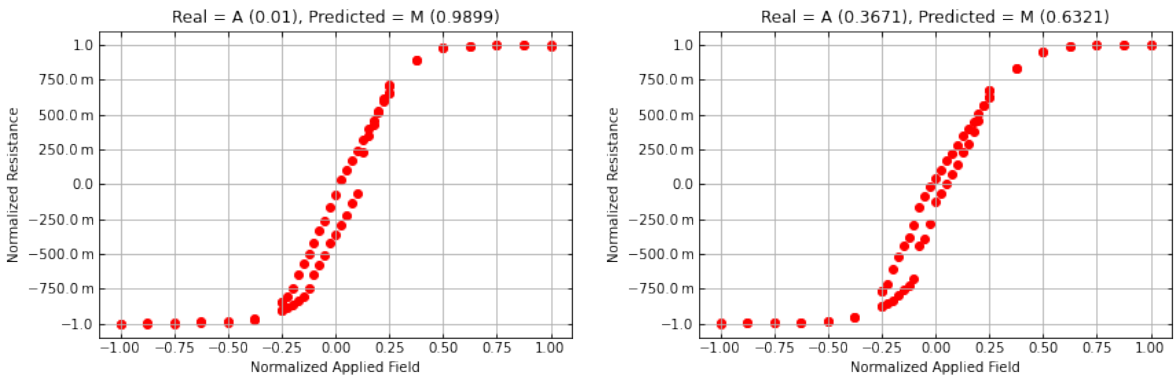


Figure 3.11: Confusion matrix heatmap for the final model evaluation using a test dataset with 30% of all data.

Fig. 3.11 shows the Confusion Matrix heatmap of the final model performance over the test dataset. This type of map helps to evaluate the performance of the model over all classes. In this map each entry with coordinates (i, j) represents the number of observations known as being from group j and predicted by the model to be as the group i. By looking at Fig. 3.11 one sees that the model over classifies curves as class "M". This might result from the fact that there is a minimal difference between this class and some curves of the "A" class and some of the "OK" class. If this class was to be removed, I expect that the model's performance would improve; however, I consider that this class is useful when trying to scavenge all available fabricated sensors.

In this final model (with a larger test dataset), the accuracy over the test dataset was 0.9672, and the F_1 score was 0.9257, giving thus good results. I expect the model to give similar results in upcoming wafers as long as some requirements are fulfilled:

- The low resistance plateau must be on the left-hand side of the curve and the high resistance state on the right-hand side. If needed this can be achieved by simply mirroring the curves;
- If the curves have a number of sampling points different from 70, either upsampling or downsampling needs to be performed.



(a) "A" curve predicted has a "M".

(b) "A" curve predicted as a "M".

Figure 3.12: Normalized $R(H)$ plots with failed prediction made by the model. Probabilities can be seen in the plot title.

3.2.2.6 Interface

Again I made an easy to use interface to make automatic classification of sensors. Users must be aware that this classifier's only considers the shape of the curve and not any quantitative parameters, such as MR. It should only be used as a complementary information. The output of this function is an excel file with predictions on the curves submitted to the model.

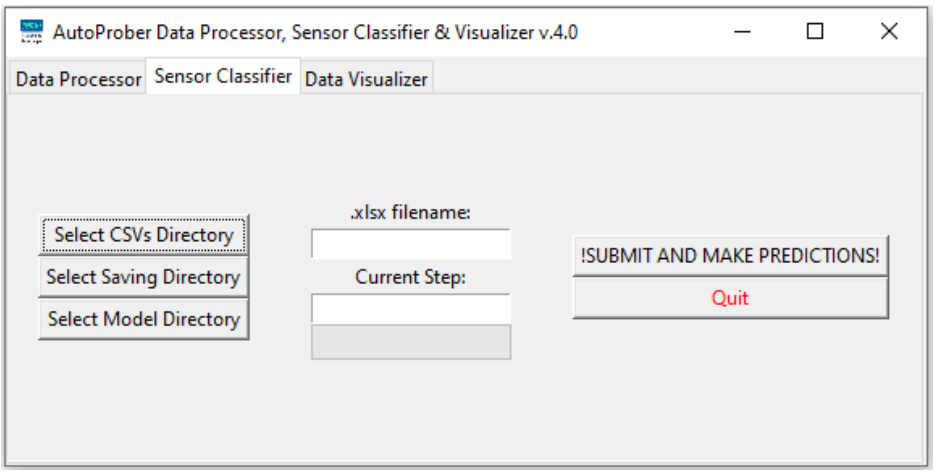


Figure 3.13: Autoprober Sensor Classifier version 4.0.

In order for the script to successfully run, the user has to input the following data:

- **"Select CSVs Directory"** - The user should select the CSVs to be processed; the CSVs should be the ones which are stored inside the **R_H_curves/SMP/SMP Name/Data** or at least files with the same structure;
- **"Select Saving Directory"** - The user should select the main directory where he wishes to save the predictions file. Bear in mind that the actual prediction file is saved under the path **"Saving Directory/Predictions/"**;
- **"Select Model Directory"** - The user can select the trained model to be used. The user should

select the folder with the model inside, and it should be saved under the TensorFlow SavedModel format. The final model trained above is supplied alongside the code;

- **".xslx filename"** - The user should write the name of the prediction file to be created without the ".xslx" extension; the extension is automatically placed. Tab 3.10 shows an example of an typical output of such routine.

Map Index	Prediction	OK	M	A	NOK
0	A	5.1E-07	0.000293	0.992958	0.006748
1	M	0.305614	0.692587	0.00177	2.89E-05
2	M	0.491555	0.501679	0.006718	4.76E-05
...	...	...	...	...	...

Table 3.10: Small portion of an example excel file generated by the Sensor Classification routine. The user may see the prediction for each of the sensors and the probabilities of each class, the class with the highest probability corresponds to the prediction.

The result of running such a classifier is an excel file analogous to what is seen on the Tab. 3.10.

3.2.3 Data Visualizer

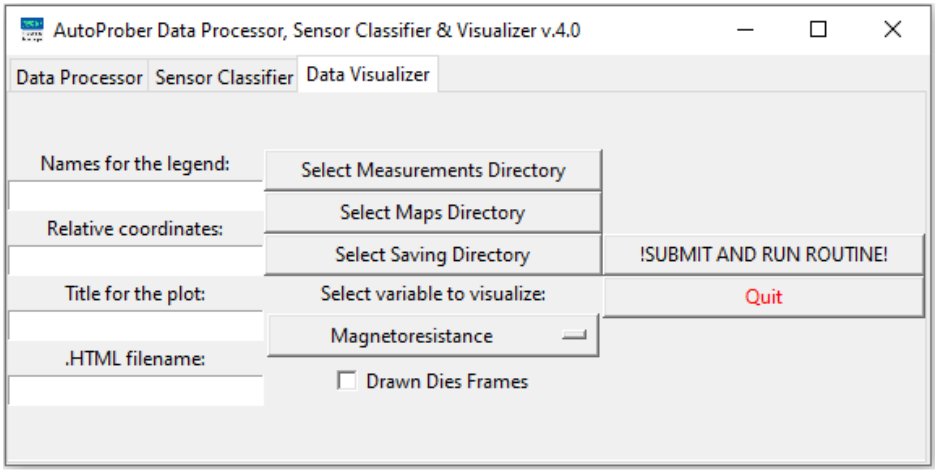


Figure 3.14: Autoprober Data Visualizer version 4.0 with the default values.

The Data Visualizer uses files computed by the Data Processor/Sensor Classifier, and the map used in the Autoprober; with this information, it does an interactive plot which is saved in ".HTML" format, and hence it can be opened in a wide variety of platforms. In Fig. 3.15, one may check a representative output, in this case, the mapping of the MR across an example sample with 16 columns of sensors in the x direction and 17 rows in the y direction.

In order for the script to successfully run, the user has to input the following data:

- **"Names for the legend:"** - The user should input the titles for the points separated by commas, e.g., $MR1$, $MR2$. If there is only one quantity, the user should input only one word, no comma;

- **"Relative coordinates:"** - This is a rather important parameter; it represents the static translation that the map coordinates should suffer with respect to the coordinate system of the final plot

generated. If the measurements map is ((0,0), (1,1), ..., (n, n)) and one wants the information to be drawn on the coordinates ((1,1), (2,2), ..., $(n+1, n+1)$) therefore the user should input the coordinates (1, 1) in the entry field. This is of special importance when one has multiple information to be plotted in the same interactive plot. Sometimes one may use the same map for different measurements across a wafer; the only difference is the initial position of the Autoprober probes. If this parameter did not exist, all the information would overlap in the final plot. The user should always input the relative coordinates even if they are (0,0);

- **"Title for the plot:"** - Title to be shown in the plot;
- **".HTML filename:"** - The filename of the .HTML file, the extension is automatically placed by the program;
- **"Select Measurements Directory"** - Select the folder with the measurements that one wishes to display;
- **"Select Maps Directory"** - Select the folder which has the maps inside;
- **"Select Saving Directory"** - Select the saving directory where the .HTML file will be saved;
- **"Select variable to visualize:"** - Select from the list the variable that he is trying to visualize;
- **"Drawn Dies Frames"** - If the user selects this function, some windows pop up, asking the die dimension and how many are in the x and y-direction.

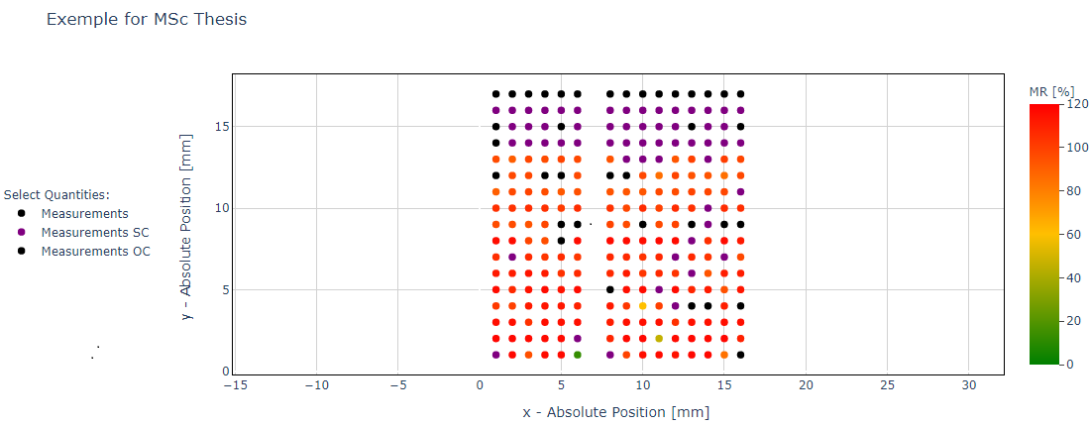


Figure 3.15: Autoprober Data Visualizer version 4.0 example using MR information. In this case the MR is plotted with OC/SC filtering.

Currently, the Autoprober Data Visualizer can plot the MR, R_{min}, the resistance at a given field, and the curve classifier's outcome.

In all examples, by hovering the mouse over the points, the user may see relevant information about the plot. For instance, in the example of the prediction, each class's probability is present, Fig. 3.16.

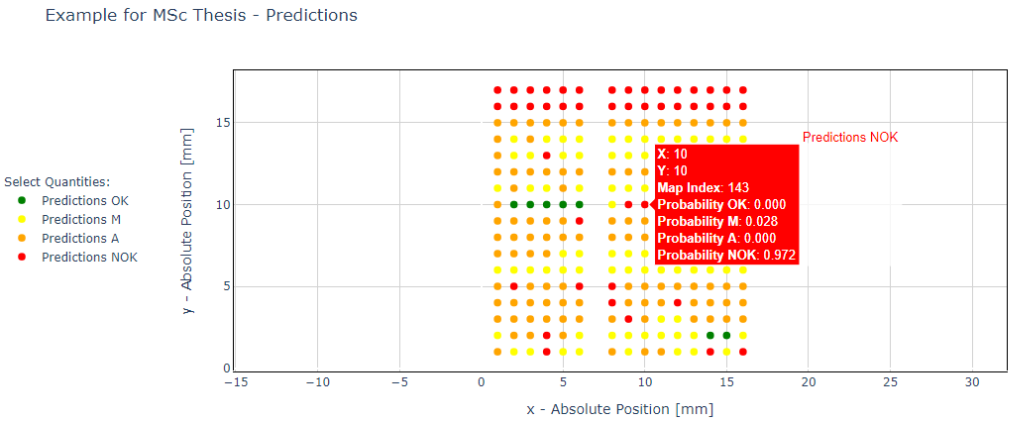


Figure 3.16: Autoprober Data Visualizer version 4.0 example using predictions from the Sensor Classifier. Mouse hover information is displayed.

3.3 Automatic Grading System

Despite having already a Machine Learning Approach to the problem, which gave satisfactory results, I tried to come up with another solution and check whether its performance is superior.

One perfect sensor has no hysteretic behavior, meaning a unique correspondence between resistance and magnetic field. With this in mind and knowing that most $R(H)$ are made with symmetric magnetic fields applied, it is possible to create a grading system that checks if the values of resistance are close to each other at the same magnetic field.

After computing all distances it terms of resistance at each magnetic field value for each curve, a normalization is done, and all distances are normalized between 0 and 1. 0 being the most favorable case and 1 being the curve with the highest distance between the branches of the hysteresis loop.

Having a total distance of 0 would mean that the two branches of the hysteresis totally overlap.

I based this idea in the hypothesis that different threshold grading values would clearly separate all the different classes; if this happened, it would be easy to separate all the different classes. The user would need to open the plots to define the threshold values of where each class starts, but that would be easy to do.

The core functions to implement such an algorithm may be seen in Appendix A.2.

To test this theory, I used the algorithm in the dataset that I already had classified for training and testing the Machine Learning Approach. In the Tab. 3.11, one may see each class's number across different ranking intervals. As one can see, it is impossible to separate the data by a ranking/grade threshold; therefore, this approach to the problem is not feasible. However, it is clear that there are some clusters of classes, and the succession of the ranking is what one would expect the **NOK** class, which is spread almost in all intervals, which makes impossible the separation by finite thresholds.

In Fig. 3.17, one may see a graphical distribution of the classes across the ranking. It is evident the presence of the **NOK** class on almost every ranking.

As the results did not show any good advantage over the Machine Learning model, I did not improve this approach and kept the Machine Learning one.

		Class distribution				
		#OK	#M	#A	#NOK	Sum
Ranking interval	[0; 62[	62	0	0	0	62
	[62; 124[	27	0	0	35	62
	[124; 186[	6	5	0	51	62
	[186; 248[	2	22	11	27	62
	[248; 310[	0	22	24	16	62
	[310; 372[	1	17	23	21	62
	[372; 434[	0	7	39	16	62
	[434; 496[	0	0	42	20	62
	[496; 558[	0	0	22	40	62
	[558; 620[	0	0	15	47	62
	[620; 682[	0	0	14	48	62
	[682; 744[	0	0	9	53	62
	[744; 806[	0	0	17	45	62
	[806; 868[	0	0	17	45	62
	[868; 930[	0	0	28	34	62
	[930; 992[	0	0	29	33	62
	[992; 1054[	0	0	15	47	62
	[1054; 1114[	0	0	20	40	60

Table 3.11: Number of occurrences of each class per ranking interval.

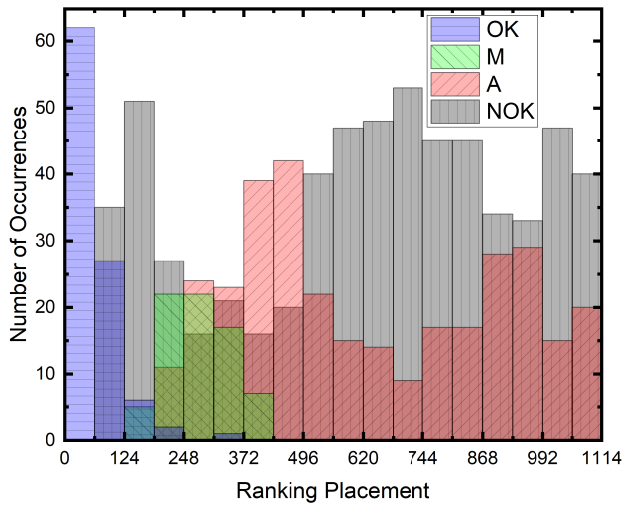


Figure 3.17: Cumulative number of occurrences of each class vs. ranking placement.

3.4 Autoprober Data Processor, Sensor Classifier & Visualizer v4.0 Installation Instructions

In the Ref. [43], one may find all the necessary files of the Autoprober Data Processor, Sensor Classifier & Visualizer. Inside the Ref. [43], there is a ”ReadMe.txt” file; if one needs further guidance, please read it.

To easily use the code, I recommend installing Anaconda as it includes almost all necessary packages used by the code. I recommend following the steps (internet connections is required):

1. Install Anaconda (Python 3), Ref. [38];
2. Install Plotly, Ref. [41]. One can open Anaconda Prompt and execute ”conda install -c anaconda plotly”;
3. If the user also wishes to use the Sensor Classifier, then he should follow the steps (a) and (b); otherwise, skip to item 4;
   (a) If the computer has an Nvidia graphics processing unit (GPU), I recommend installing Keras/Tensorflow with GPU support. One can open Anaconda Prompt and execute ”conda install -c anaconda keras-gpu”;
   (b) If the computer does not have an Nvidia GPU, one should install Keras/Tensorflow central processing unit based. One can open Anaconda Prompt and execute ”conda install -c anaconda keras”;
4. Open Jupyter Notebook, and one may check if every package is installed by trying to import it[2].

After the installation, one may open the software and use it. One finds ”.py” and ”.ipynb” files in the Ref. [43]. Choose whatever integrated development environment one wishes, but I recommend using the Jupyter Notebook and hence the ”.ipynb” file. All the required packages are open-source.

[2]This is not mandatory but advised as it lets users check if every required package is installed.

Chapter 4

Proposal for Predicting Sensor Fabrication Yield

This is a very challenging topic; I will, in this chapter, present a possible approach to predict sensor yield based on the process itself. As already stated in Sec. 2.2.1, and in particular in Tab. 2.2, there are many challenges to the fabrication of a MR sensor.

The approach hereby presented is Machine Learning based.

4.1 Data Gathering

One of the most challenging parts of Machine Learning is understanding what data is relevant or not. Since this is a novel problem, one should first define what data should be collected before proceeding.

When asking someone for that, one should try to facilitate the executor's job. For instance, if one develops a process characterization runsheet to be filled alongside the regular runsheet, it is evident that such a document should be fast and easy to fill. If the document is dull, it may fail to engage users in filling it.

For the beginning stage, I propose starting only with the analysis of lithography and etching steps. With this in mind, I developed the runsheet present in Appendix B. To fill in the information asked on the file, the users must first understand Tabs. 4.1 and 4.2. It is of most importance to keep consistency between attributed values to different sensors and steps.

		Encoding
Description	Any defect that **for sure** puts at risk the operability of the device. E.g., non-defined structure, totally wrong defined structure.	NOK
	Any defect that **may** put at risk the device's response but not in a drastic way. E.g., *lics* misalignment, small defects.	M
	Perfectly defined structure, or structures with minimal defects that will not affect the device's final performance notably.	OK
	No information.	NA

Table 4.1: Encoding table for filing the Appendix B lithography steps tables.

		Encoding
Description	Sensors suffered underetching?	UE
	Sensors suffered overetching?	OE
	Sensors were etched accordingly.	OK
	No information.	NA

Table 4.2: Encoding table for filing the Appendix B etching steps tables.

The etching steps may sometimes be difficult to evaluate; this arises from the fact that some layers are so thin that it is difficult to know if one has etched the sample as supposed or not. Sometimes, one can only rely on optical inspection and check the contrast of colors, as different materials have different colors. When the layers are thick, one can rely on the profilometry to evaluate the etch depth.

Considering what was explained in Sec. 2.2.1, many processes have radial symmetry, and therefore characterization can be made symmetrically for such cases. In all steps of the runsheet present in Appendix B, the user should take notes of structures that are defective as well as structures that do not have any problem; more details are available on the document itself.

After a defect is detected on a structure, it should be followed up in the rest of the process until its final characterization.

The file in the Appendix B was created for MTJ based process; however, it can be adapted to any process available at INESC MN.

I am not sure of the amount of data needed for this model's results to be satisfactory, but a rough figure can be calculated.

If all classes have equal probability and if eight measurements are taken in the first lithography and two on upcoming steps, then[1]:

$$\text{\# Possibilities} = 4^8 = 65536$$

$$\text{\# Possibilities tested per run} = 8 \times 2^7 = 1024$$

$$\text{\# Runs needed in case of equal probability of classes} = \frac{65536}{1024} = 64$$

I think that sixty-four is a pretty unrealistic number as some classes will for sure be imbalanced. Some of this data could be artificially created; if everything goes wrong, it is evident that the sensor will not work no matter the sensor's placement, and this reasoning could be made for other situations.

I would guess that $10 \times 64 = 640$ samples processed would be more than sufficient data to develop and train a good classifier.

[1] This estimation was made for the standard process that includes five lithography steps and three etchings.

4.2 Data Treatment

After gathering the data, it should be organized to train the model successfully.

Each process step's evaluation should be a feature, and the target should be the device's classification. One should use all the available outputs of Autoprober Data Processor, Sensor Classifier & Visualizer before deciding on the sensor classification. The Sensor Classifier could be used to check the shape of the curve and another complementary information to check if the sensor is within expected values, for instance, the MR or R_{min}.

Features						...
x [mm]	**y [mm]**	**1st Lithography (Bottom Electrode)**	**1st Etch (Bottom Electrode)**	**2nd Lithography (Pillar)**	**2nd Etch (Pillar)**	...
0	0	OK	OK	OK	OK	...
100	100	NOK	OE	NOK	OE	...

...	**Features**				**Targets**
...	**3rd Lithography (Vias)**	**3rd Etch (Vias)**	**4th Lithography (Top Electrode)**	**5th Lithography (Passivation)**	**Sensor Classification**
...	OK	OK	OK	OK	OK
...	NOK	OE	NOK	NOK	NOK

Table 4.3: Example of what data I would use to train a supervised model to predict sensor yield.

The entries on the final data table should all use the same coordinate system, and when the users use the model to make predictions, they should use the same coordinate system. The features, except for the coordinates, should be one-hot encoded.

4.3 Model Training, Limitations and Expected Results

Regarding the model selection, I would use the same approach as I used in Sec. 3.2.2.3. I would again try different solutions and check which one had better performance over a validation dataset. Afterward, I would train the final model with the same architecture that gave rise to better results in the validation dataset.

There are some clear limitations to the approach presented, I consider two very important:

- If the process changes, *i.e.* the number of steps or even the order, then one should use another model and training;
- The stack is not included as a feature. This might result in poor predictions from the model since the stack composition plays a significant role in the device itself.

If the model is appropriately trained, it will be advantageous to evaluate, for instance, exclusion zones of productions, which allows saving time. However, I insist that somehow the stack should also be one feature. Maybe even as a simple string, which could be one-hot encoded.

With this approach, the users can then understand which steps are the most critical of a micro or nanofabrication process and then try to improve them.

I cannot elaborate more on this theme since I do not have real examples; as I already stated, this is only a proposal of how to predict the yield of a given process.

Chapter 5

Conclusions and Future Work

In this chapter one may find a brief conclusions about the work done and what I propose to be the future developments of these tools.

5.1 Work Accomplished

With this Master Thesis, I have created an integrated and easy to use tool that can provide a fast quantitative analysis of a fabricated sample. The tool includes an automatic curve classifier that can successfully distinguish between what is an appropriate MR sensor response and what is not. The conjugation of the output of the quantitative analysis and the classifier is enough to make an automatic qualification of the sensors which are within working parameters.

After testing the classifier on different datasets, I checked that it performs as expected both on SV and MTJs based sensors. I firmly believe that the classifier will classify well $>$90% of all curves; the only problem that I expect to happen is an over-classification of curves as the class ”M”.

Since I used Python to develop such a tool, it can be used in a broad spectrum of platforms. It is easy to install and easy to use, fulfilling all the users' requirements at INESC MN.

On top of that, I was also able to create a tool to visualize the results, which makes two-dimensional scatter plots with hover information on each point. It allows the filtering of different types of data and results quickly. When applicable, a continuous color scale is available, for instance, when plotting MR. All the plots are interactive and saved in a format that can be open by any ”.HTML” reader, for example, any web browser.

After this work, I tackled the last main challenge of my Thesis, proposing a technique to try to predict the yield of a micro or nanofabrication process to identify the process's critical steps. In this topic, I successfully created a Process Characterization Runsheet, which allows gathering the necessary data to accomplish this task. The Runsheet created is for an MTJ based process, but it can be adapted for SV.

In this ambit, I also proposed what kind of treatment should be done to the data and what kind of models I would try to train to predict the desired parameters, in this case, the yield.

If such approximation is implemented and the users use it to predict the yield, I recommend that they look at each class's probability and not only at the prediction itself. It goes without saying that having a class predicted with a very high probability is totally different from having it predicted with a very low probability hence the need to look at all the available information.

5.2 Future Work

Regarding the first and second objectives of this Master Thesis and the solution implemented I have the following recommendations/suggestions:

Due to the modular nature in which I built the Data Processor more analysis functions could be added to calculate other relevant parameters such as for instance the sensitivity.

I encourage users to tailor the code to their needs if needed.

Maybe for the future, the code could be placed on a central service available through the internet. In this way, users could process their files without the need to have the code on their computers. In spite of not being a goal of this Thesis, I am now adapting the code to meet this goal. For this, I am using Google Colab, which supports Jupyter Notebooks. One may see the progress inside the drive in the folder "Cloud-based". Ref. [43] Please bear in mind that this is a work in progress and was not a direct goal of this Thesis, some adaptation is needed because it is impossible to do a GUI using this platform.

Regarding what was the third objective of this Thesis, the proposal is completed; however, to gather the required data for the implementation of such a model successfully, there is the need to try and engage the users at INESC MN to fill the Process Characterization Runsheet.

Bibliography

[1] C. Zheng, K. Zhu, S. Cardoso de Freitas, J. Chang, J. E. Davies, P. Eames, P. P. Freitas, O. Kazakova, C. Kim, C. Leung, S. Liou, A. Ognev, S. N. Piramanayagam, P. Ripka, A. Samardak, K. Shin, S. Tong, M. Tung, S. X. Wang, S. Xue, X. Yin, and P. W. T. Pong. Magnetoresistive Sensor Development Roadmap (Non-Recording Applications). *IEEE Transactions on Magnetics*, 55(4):1–30, April 2019. ISSN 1941-0069. doi: 10.1109/TMAG.2019.2896036.

[2] T. Barroso, R. Martins, E. Fernandes, S. Cardoso, J. Rivas, and P. P. Freitas. Detection of BCG bacteria using a magnetoresistive biosensor: A step towards a fully electronic platform for tuberculosis point-of-care detection. *Biosensors and Bioelectronics*, 100, 09 2017. doi: 10.1016/j.bios.2017.09.004.

[3] X.-H. Mu, H.-F. Liu, Z.-Y. Tong, B. Du, S. Liu, B. Liu, Z.-W. Liu, C. Gao, J. Wang, and H. Dong. A new rapid detection method for ricin based on tunneling magnetoresistance biosensor. *Sensors and Actuators B: Chemical*, 284:638 – 649, 2019. ISSN 0925-4005. doi: https://doi.org/10.1016/j.snb.2018.12.127. URL `http://www.sciencedirect.com/science/article/pii/S0925400518322470`.

[4] B. Yang and Y. Lei. Vehicle Detection and Classification for Low-Speed Congested Traffic With Anisotropic Magnetoresistive Sensor. *IEEE Sensors Journal*, 15(2):1132–1138, Feb 2015. ISSN 2379-9153. doi: 10.1109/JSEN.2014.2359014.

[5] P. Pai, L. Chen, F. K. Chowdhury, and M. Tabib-Azar. Non-intrusive electric power sensors for smart grid. In *SENSORS, 2012 IEEE*, pages 1–4, 2012.

[6] P. P. Freitas, R. Ferreira, S. Cardoso, and F. Cardoso. Magnetoresistive sensors. *Journal of Physics: Condensed Matter*, 19(16):165221, April 2007. doi: 10.1088/0953-8984/19/16/165221.

[7] P. P. Freitas, R. Ferreira, and S. Cardoso. Spintronic Sensors. *Proceedings of the IEEE*, 104(10): 1894–1918, 2016.

[8] Magnetic Measurement - Bogen. Bogen Electronic GmbH, 2020. URL `http://www.bogen-electronic.com/en/magnetic-measurement-solutions/technology/magnetic-measurement.html`. Accessed on 27 August 2020.

[9] Bosch Sensortec: Bosch MEMS Technology. Bosch Sensortec GmbH, 2020. URL `https://www.bosch-sensortec.com/`. Accessed on 27 August 2020.

[10] J. Plummer, M. Deal, and P. Griffin. *Silicon VLSI Technology: Fundamentals, Practice and Modeling*. Prentice Hall, 07 2000. ISBN 978-0-1308-5037-9.

[11] W. Thomson. XIX. On the electro-dynamic qualities of metals:-Effects of magnetization on the electric conductivity of nickel and of iron. *Proceedings of the Royal Society of London*, 8:546–550, 1857. doi: 10.1098/rspl.1856.0144. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspl.1856.0144.

[12] M. N. Baibich, J. M. Broto, A. Fert, F. N. Van Dau, F. Petroff, P. Etienne, G. Creuzet, A. Friederich, and J. Chazelas. Giant Magnetoresistance of (001)Fe/(001)Cr Magnetic Superlattices. *Phys. Rev. Lett.*, 61:2472–2475, Nov 1988. doi: 10.1103/PhysRevLett.61.2472. URL https://link.aps.org/doi/10.1103/PhysRevLett.61.2472.

[13] G. Binasch, P. Grünberg, F. Saurenbach, and W. Zinn. Enhanced magnetoresistance in layered magnetic structures with antiferromagnetic interlayer exchange. *Phys. Rev. B*, 39:4828–4830, Mar 1989. doi: 10.1103/PhysRevB.39.4828. URL https://link.aps.org/doi/10.1103/PhysRevB.39.4828.

[14] The Nobel Prize in Physics 2007. *NobelPrize.org*, 2007. URL https://www.nobelprize.org/prizes/physics/2007/summary/. Accessed on 07 July 2020.

[15] K. M. Krishnan. *Fundamentals and Applications of Magnetic Materials*. Oxford University Press, Oxford, 2016. ISBN 9780199570447. doi: 10.1093/acprof:oso/9780199570447.001.0001. URL https://global.oup.com/academic/product/fundamentals-and-applications-of-magnetic-materials-9780199570447.

[16] T. Miyazaki and N. Tezuka. Giant magnetic tunneling effect in Fe/Al_2O_3/Fe junction. *Journal of Magnetism and Magnetic Materials*, 139(3):L231 – L234, 1995. ISSN 0304-8853. doi: https://doi.org/10.1016/0304-8853(95)90001-2. URL http://www.sciencedirect.com/science/article/pii/0304885395900012.

[17] J. S. Moodera, L. R. Kinder, T. M. Wong, and R. Meservey. Large Magnetoresistance at Room Temperature in Ferromagnetic Thin Film Tunnel Junctions. *Phys. Rev. Lett.*, 74:3273–3276, Apr 1995. doi: 10.1103/PhysRevLett.74.3273. URL https://link.aps.org/doi/10.1103/PhysRevLett.74.3273.

[18] M. Julliere. Tunneling between ferromagnetic films. *Physics Letters A*, 54(3):225 – 226, 1975. ISSN 0375-9601. doi: https://doi.org/10.1016/0375-9601(75)90174-7. URL http://www.sciencedirect.com/science/article/pii/0375960175901747.

[19] H. Zheng, J. Fu, T. Mei, and J. Luo. Learning Multi-attention Convolutional Neural Network for Fine-Grained Image Recognition. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5219–5227, 2017.

[20] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

[21] A. Borovykh, S. Bohte, and C. W. Oosterlee. Conditional Time Series Forecasting with Convolutional Neural Networks, 2017.

[22] C. Barata and J. S. Marques. Deep Learning For Skin Cancer Diagnosis With Hierarchical Architectures. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 841–845, 2019.

[23] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL `https://doi.org/10.1007/BF02478259`.

[24] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[25] S. Raschka and V. Mirjalili. *Python Machine Learning, 3rd Ed.* Packt Publishing, Birmingham, UK, 2019. ISBN 978-1789955750.

[26] K. Seshan. *Handbook of Thin Film Deposition Processes and Techniques*. William Andrew Publishing, Norwich, NY, second edition edition, 2002. ISBN 978-0-8155-1442-8.

[27] K. Seshan and D. Schepis. *Handbook of Thin Film Deposition*. William Andrew Publishing, Oxford, fourth edition edition, 2018. ISBN 978-0-1281-2311-9.

[28] A. V. Hayes, V. Kanarov, R. Yevtukhov, H. Hegde, B. Druz, D. Yakovlevitch, W. Cheesman, and V. Mirkov. Ion source for ion beam deposition employing a novel electrode assembly. *Review of Scientific Instruments*, 71(2):1163–1167, 2000. doi: 10.1063/1.1150416.

[29] R. Waser. *Nanoelectronics and Information Technology*. Wiley-VCH, 3rd edition, 2012. ISBN 978-3-5274-0927-3.

[30] B. Päivänranta, A. Langner, E. Kirk, C. David, and Y. Ekinci. Sub-10 nm patterning using EUV interference lithography. *Nanotechnology*, 22(37):375302, August 2011. doi: 10.1088/0957-4484/22/37/375302. URL `https://doi.org/10.1088%2F0957-4484%2F22%2F37%2F375302`.

[31] S. Reyntjens and R. Puers. A review of focused ion beam applications in microsystem technology. *Journal of Micromechanics and Microengineering*, 11(4):287–300, jul 2001. doi: 10.1088/0960-1317/11/4/301. URL `https://doi.org/10.1088/0960-1317/11/4/301`.

[32] A. V. Silva. *Towards sub-100nm Magnetoresistive Devices: From Simulations to Applications*. PhD thesis, Universidade de Lisboa - Instituto Superior Técnico, 2016.

[33] S. Gnanarajan. Using masks to obtain uniform ion etch rates. *Review of Scientific Instruments*, 73(4):1853–1855, 2002. doi: 10.1063/1.1464657.

[34] N. Savvides. Correction masks for large-area ion beam etching and figuring of optics. *Journal of Applied Physics*, 99(9):094912, 2006. doi: 10.1063/1.2197035.

[35] S. Matsuo and Y. Adachi. Reactive Ion Beam Etching Using a Broad beam ECR Ion Source. *Japanese Journal of Applied Physics*, 21(Part 2, No. 1):L4–L6, jan 1982. doi: 10.1143/jjap.21.l4.

[36] S. Franssila and L. Sainiemi. *Reactive Ion Etching (RIE)*, pages 1772–1781. Springer US, Boston, MA, 2008. ISBN 978-0-387-48998-8. doi: 10.1007/978-0-387-48998-8_1344. URL https://doi.org/10.1007/978-0-387-48998-8_1344.

[37] Python Software Foundation. Python Language Reference, version 3.8, 2020. URL https://www.python.org/. Accessed on 12 September 2020.

[38] Anaconda software distribution. *Anaconda Documentation*, 2020. URL https://docs.anaconda.com/. Accessed on 18 September 2020.

[39] F. Chollet et al. Keras. *GitHub repository*, 2015. URL https://keras.io. Accessed on 20 September 2020.

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[41] Plotly Technologies Inc. *Collaborative data science*, 2015. URL https://plot.ly.

[42] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[43] D. Sousa. Autoprober Data Processor, Sensor Classifier & Visualizer - Google Drive. 2020. URL https://drive.google.com/drive/folders/1RgHbQ2UwFniroyRiAt67cKYSTcLIhd-u?usp=sharing. Accessed on 30 October 2020.

[44] Y. LeCun, C. Cortes, and C. J.C. Burges. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/. Accessed on 17 September 2020.

[45] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

Appendices

Appendix A - Code

As stated previously in the document, the full extent of the code is not available on this document due to its high amount of lines, however some important functions are available here. For easier reading the functions have comments.

A.1 Training and Testing Models

```
#!/usr/bin/env python
# coding: utf-8

# # Sensor classification - Training models

# ## Imports

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from keras.optimizers import Adam
from keras.models import Sequential, load_model
from keras.losses import categorical_crossentropy
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
from keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D, Dropout


# ## Loading datasets

X_train = np.load('Data Final/Datasets/X_train.npy')
X_val = np.load('Data Final/Datasets/X_val.npy')
X_test = np.load('Data Final/Datasets/X_test.npy')

y_train = np.load('Data Final/Datasets/y_train.npy')
y_val = np.load('Data Final/Datasets/y_val.npy')
y_test = np.load('Data Final/Datasets/y_test.npy')


# ## Models

# ### Model selection

model_selection = '8' #CHOOSE HERE!

# ### Model 1

if model_selection == '1':
    print('Model 1 activated.')
```

```
    model = Sequential()
    model.add(Conv2D(32, (3, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((2, 1)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 2

if model_selection == '2':
    print('Model 2 activated.')
    model = Sequential()
    model.add(Conv2D(32, (7, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((5, 1)))
    model.add(Flatten())
    model.add(Dense(50, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 3

if model_selection == '3':
    print('Model 3 activated.')
    model = Sequential()
    model.add(Conv2D(16, (7, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((5, 1)))
    model.add(Flatten())
    model.add(Dense(50, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 4

if model_selection == '4':
    print('Model 4 activated.')
    model = Sequential()
    model.add(Conv2D(16, (7, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((5, 1)))
    model.add(Conv2D(16, (3, 1), activation='relu'))
    model.add(MaxPooling2D((2, 1)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 5

if model_selection == '5':
    print('Model 5 activated.')
    model = Sequential()
    model.add(Conv2D(16, (7, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((5, 1)))
```

```
    model.add(Dropout(0.4))
    model.add(Conv2D(16, (3, 1), activation='relu'))
    model.add(MaxPooling2D((2, 1)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 6

if model_selection == '6':
    print('Model 6 activated.')
    model = Sequential()
    model.add(Conv2D(16, (7, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((5, 1)))
    model.add(Dropout(0.4))
    model.add(Conv2D(16, (3, 1), activation='relu'))
    model.add(MaxPooling2D((2, 1)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 7

if model_selection == '7':
    print('Model 7 activated.')
    model = Sequential()
    model.add(Conv2D(16, (7, 2), activation='relu', input_shape=(70, 2, 1)))
    model.add(MaxPooling2D((5, 1)))
    model.add(Conv2D(16, (3, 1), activation='relu'))
    model.add(MaxPooling2D((2, 1)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(4, activation='softmax'))

# ### Model 8

if model_selection == '8':
    print('Model 8 activated.')
    model = Sequential()
    model.add(Dense(100, activation='relu', input_shape=(70, 2, 1)))
    model.add(Dropout(0.4))
    model.add(Dense(50, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(4, activation='softmax'))


```

```
# ## Compilation, early stopping and fitting

print(model.summary())

model.compile(optimizer = Adam(), loss = categorical_crossentropy,
              metrics = ['accuracy'])

es = EarlyStopping(monitor = 'val_loss', mode='min', verbose = 1, patience = 50)
mc = ModelCheckpoint('Models/best_model_' + model_selection + '.h5',
                     monitor = 'val_accuracy', mode = 'max',
                     verbose = 1, save_best_only = True)

history = model.fit(X_train, y_train, epochs = 1000,
                    validation_data=(X_val, y_val), callbacks = [es, mc])


# ## Model evaluation

plt.plot(history.history['loss'], label='Train dataset')
plt.plot(history.history['val_loss'], label='Validation dataset')
plt.grid()
ax = plt.gca()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss vs. epoch in the train and validation dataset\nModel ' + model_selection)
plt.legend()
plt.tight_layout()
plt.savefig('Models/Loss vs. epoch/model_' + model_selection + '.png')
plt.show()
print('The best model ' + model_selection + ' was trained in the epoch ' +
      str(history.history['val_accuracy'].index(max(history.history['val_accuracy']))) +
      '.')

df_loss = pd.DataFrame({'Epoch':range(len(history.history['loss'])),
                   'Loss in train':history.history['loss'],
                   'Loss in validation':history.history['val_loss']})
df_loss.to_excel('Models/Loss vs. epoch/model_' + model_selection + '.xlsx',
                 index = False)

def decoder(y_array_encoded):
    decoded_list = []
    for i in range(len(y_array_encoded)):
        if np.array_equal(y_array_encoded[i], np.array([1, 0, 0, 0])):
            l = 'OK'
        elif np.array_equal(y_array_encoded[i], np.array([0, 1, 0, 0])):
            l = 'M'
        elif np.array_equal(y_array_encoded[i], np.array([0, 0, 1, 0])):
            l = 'A'
        elif np.array_equal(y_array_encoded[i], np.array([0, 0, 0, 1])):
```

```
            l = 'NOK'
        decoded_list.append(l)
    decoded_array = np.array(decoded_list)
    return decoded_array

def predict_proba_to_oneh(predict_prob):
    encoded_array = np.empty((0, 4), int)
    for i in predict_prob:
        if i[0] > i[1] and i[0] > i[2] and i[0] > i[3]:
            l = np.array([[1, 0, 0, 0]])
        elif i[1] > i[0] and i[1] > i[2] and i[1] > i[3]:
            l = np.array([[0, 1, 0, 0]])
        elif i[2] > i[0] and i[2] > i[1] and i[2] > i[3]:
            l = np.array([[0, 0, 1, 0]])
        elif i[3] > i[0] and i[3] > i[1] and i[3] > i[2]:
            l = np.array([[0, 0, 0, 1]])
        else:
            print('Function must be wrong.')
        encoded_array = np.append(encoded_array, l, axis=0)
    return encoded_array

model = load_model('Models/best_model_' + model_selection + '.h5')
y_pred_oneh_test = predict_proba_to_oneh(model.predict(X_test))
y_pred_test_decoded = decoder(y_pred_oneh_test)
y_test_decoded = decoder(y_test)
prob_test = model.predict(X_test)


print('Accuracy Score of the model ' + model_selection + ': ' +
      str(round(accuracy_score(y_test_decoded, y_pred_test_decoded), 4)))
print('F1 Score of the model ' + model_selection + ': ' +
      str(round(f1_score(y_test_decoded, y_pred_test_decoded, average = 'macro'), 4)))


data = {'y_Actual':    y_test_decoded,
        'y_Predicted': y_pred_test_decoded
        }

df = pd.DataFrame(data, columns=['y_Actual','y_Predicted'])

confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'],
                               rownames=['Actual'], colnames=['Predicted'])

ax = sns.heatmap(confusion_matrix, annot = True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()

print('The best model ' + model_selection + ' was trained in the epoch ' +
```

```
      str(history.history['val_accuracy'].index(max(history.history['val_accuracy']))) +
      '.')


# ## Cheking what failed

def get_prob_real(real_label, prob_array):
    if real_label == 'OK':
        prob = prob_array[0]
    elif real_label == 'M':
        prob = prob_array[1]
    elif real_label == 'A':
        prob = prob_array[2]
    elif real_label == 'NOK':
        prob = prob_array[3]
    else:
        print('Please check the real label, such label is impossible.')
    return prob

plot_x_failed = []
pred_x_failed = []
real_x_failed = []
prob_pred_x_failed = []
prob_real_x_failed = []
for i in range(len(y_test_decoded)):
    if y_pred_test_decoded[i] != y_test_decoded[i]:
        print('Predicted:', y_pred_test_decoded[i],
              'Real:', y_test_decoded[i],
              'Probability array:', prob_test[i])
        prob_real_x_failed.append(get_prob_real(y_test_decoded[i], prob_test[i]))
        plot_x_failed.append(X_test[i])
        pred_x_failed.append(y_pred_test_decoded[i])
        real_x_failed.append(y_test_decoded[i])
        prob_pred_x_failed.append(max(prob_test[i]))

plot_x_failed = np.array(plot_x_failed)
```

A.2 Grading Functions

```python
def makes_relative_distances_list(list_dfs_treated):
    '''
    Requires: list_dfs_treated must be a list containing df already resized
    Ensures:  outputs a list of distances in terms of total resistance
              for each curve
    '''
    distances_list = []
    for df in list_dfs_treated: #for each curve in all curves treated:
        list_applied_field = df['Resized H [Oe]'].tolist() #convert field to list
        list_R = df['Resized R [Ω]'].tolist() #convert resistance to list

        #applied_foward and R_foward are to be all values until the middle of each list
        #applied_backward and R_backward are to be all the remaning entries in the list
        ##applied_backawad and R_backward are reversed to be similar to the foward lists
        applied_foward = list_applied_field[:int(len(list_applied_field)/2)]
        applied_backward = list_applied_field[int(len(list_applied_field)/2):][::-1]
        R_foward = list_R[:int(len(list_R)/2)]
        R_backward = list_R[int(len(list_R)/2):][::-1]

        #This algorithm only makes senses if the same exact fields were applied
        if applied_foward == applied_backward:
            sum_distance_R = 0
            for i in range(len(applied_foward)):
                #the sum of the distance for a curve is the sum of the absolute value\
                #of distance for all points
                sum_distance_R += abs(R_foward[i] - R_backward[i])
        else:
            raise ValueError('List of fields applied is not symetric.')
        distances_list.append(sum_distance_R)
    return distances_list

def normalize_grades(distances_list):
    #get min and max of distances
    distances_list_min, distances_list_max = min(distances_list), max(distances_list)
    for i, val in enumerate(distances_list):
        #each entry of the list is now renormalized
        #x_norm = (x-min(x_list))/(max(x_list) - min(x_list))
        delta_distance = (distances_list_max-distances_list_min)
        distances_list[i] = 1.0 * (val-distances_list_min) / delta_distance
    return distances_list

normalized_distances_list = normalize_grades(distances_list)
```

Appendix B - Process Characterization Runsheet

This document was created to gather relevant data to train the model for Predicting Sensor Yield. The file only accounts for the analysis of lithography and etching steps. To fill in the Runsheet the Tabs. 4.1 and 4.2 must be consulted.

Responsible:

Process Start: ____ / ____ / _____ **Process Finish:** ____ / ____ / _____

AutoCAD Mask: **Sample:**

STEP 1: *TMR Stack Deposition* ____/____/_____

Sample ID: **Machine**

Stack:

Optical Inspection | Comments:

Sample Dimensions:

Where was the piece on the deposition chamber with respect to the substrate holder? Draw a scheme and place all relevant dimensions.

STEP 2: 1st *Lithography – Bottom Electrode Definition (L0)* ____/____/_____

Operator: **Machine:** DWL

Note: You should give at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry.

Please make a drawing of your system of coordinates with respect to the sample.

x	y	Description	Encoding

STEP 3: Bottom Electrode Definition by Ion Milling ____/____/_____

Operator: **Machine:**

Note: Please state if the etching was ok or if there were any sensors that suffered underetch or overetch

x	y	Description	Encoding

Note: Please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2 table and repeat measurements on the same coordinates.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2.

Please make a sketch of were the sample was with respect to the substrate holder, give all relevant dimensions.

STEP 4: *Resist Ashing/Strip* ____/____/_____

Operator: **Machine**

Optical Inspection | Comments:

STEP 5: 2nd *Lithography – Pillar Definition (L1)* ____/____/_____

Operator: **Machine:** DWL

Note: This is a critical step, please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2 and 3 table and repeat measurements on the same places.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2 and 3.

x	y	Description	Encoding

Optical Inspection | Comments:

STEP 6: Pillar Definition by Ion Milling ____/____/_____

Operator: **Machine:**

Note: Please state if the etching was ok or if there were any sensors that suffered underetch or overetch

x	y	Description	Encoding

Note: This is a critical step, please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2, 3 and 5 table and repeat measurements on the same places.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2, 3 and 5.

Please make a sketch of were the sample was with respect to the substrate holder, give all relevant dimensions.

STEP 7: *Resist Ashing/Strip* ____/____/_____

Operator: **Machine**

Optical Inspection | Comments:

STEP 8: Passivation ____/____/_____

Operator: **Machine:**

Comments:

STEP 9: 3rd *Lithography – Vias Opening* ____/____/_____

Operator: **Machine:** DWL

Note: This is a critical step, please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2, 3, 5 and 6 tables and repeat measurements on the same places.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2, 3, 5 and 6.

x	y	Description	Encoding

Optical Inspection | Comments:

STEP 10: *Vias Opening* ____/____/_____

Operator: **Machine:**

x	y	Description	Encoding

Note: This is a critical step, please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2, 3, 5, 6 and 9 tables and repeat measurements on the same places.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2, 3, 5, 6 and 9.

Please make a sketch of were the sample was with respect to the substrate holder, give all relevant dimensions.

STEP 11: *Resist Ashing/Strip* ____/____/_____

Operator: **Machine**

Optical Inspection | Comments:

STEP 12: 4th *Lithography – Top Electrode (contacts)* ____/____/_____

Operator: **Machine:** DWL

Note: Please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2, 3, 5, 6, 9 and 10 tables and repeat measurements on the same places.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2, 3, 5, 6, 9 and 10.

x	y	Description	Encoding

Optical Inspection | Comments:

STEP 13: Metallization ____/____/_____

Operator: **Machine:**

Comments:

STEP 14: *Metal Lift-off* ____/____/_____

Operator: **Machine**

Optical Inspection | Comments:

STEP 15: 5th *Lithography* – Passivation ____/____/_____

Operator: **Machine:** DWL

Note: Please take at least 6 measurements where everything went fine, and please take these 6 measurements with radial symmetry. If possible, check STEP 2, 3, 5, 6, 9 and 10 tables and repeat measurements on the same places.

Please make a drawing of your system of coordinates with respect to the sample, if possible, use the same coordinate system used in the STEP 2, 3, 5, 6, 9 and 10.

x	y	Description	Encoding

Optical Inspection | Comments:

STEP 16: Passivation Layer Deposition ____/____/_____

Operator: **Machine:**

Comments:

STEP 17: Lift-off ____/____/_____

Operator: **Machine:**

Comments: