Implementation of Machine Learning strategies in Resonant Ultrasound Spectroscopy

by

Felipe Giraldo Grueso

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

Bachelor of Science

in

Physics

Directed by: Dr. Paula Giraldo Gallo

Department of Physics

Universidad de Los Andes

Bogotá, Colombia

June 2021

Acknowledgements

I would like to express my deep gratitude to Professor Paula Giraldo Gallo, my research supervisor, for her patient guidance, enthusiastic encouragement, and useful critiques of this research work.

I would also like to extend my thanks to Professor Alejandro Garcia for his convenient advice during this project, to Professor Albert Migliori for providing the needed experimental data to test the deep learning solutions implemented, and, to the Department of Physics at Universidad de Los Andes for their help in offering me the resources needed to complete this work and to graduate as a physicist.

Finally, my deep and sincere gratitude to my family for their continuous love, help and support.

Resumen

La reacción elástica de un material viene dictada por su tensor de constantes elásticas. Estas constantes elásticas son una medida de las fuerzas interatómicas del material, es decir, se definen como segundas derivadas de la energía libre con respecto a la deformación en diferentes simetrías, lo que las convierte en una herramienta útil para estudiar el entorno atómico de una red cristalina. Técnicas experimentales como la espectroscopia de resonancia ultrasónica (RUS) han demostrado ser muy valiosas en la determinación de los módulos elásticos de un material. RUS usa los modos de resonancia de los cuerpos elásticos para inferir diferentes propiedades de los materiales, como sus constantes elásticos. Esta técnica experimental se divide en dos procedimientos diferentes conocidos como el problema frontal, que trata de la determinación de los modos de resonancia a partir de los módulos elásticos del cuerpo estudiado, y el problema inverso que determina los módulos elásticos a partir de las medidas experimentales de los modos de resonancia. En este proyecto se presenta una nueva aproximación al problema inverso en RUS mediante el uso de árboles de regresión en el contexto del aprendizaje automático. Los resultados obtenidos muestran que la implementación de estrategias de aprendizaje automático en RUS puede reducir la varianza de los resultados obtenidos (en comparación con las soluciones tradicionales) y puede eliminar la necesidad de tener conjeturas iniciales para los módulos elásticos al resolver el problema inverso.

Abstract

The elastic reaction of a material is dictated by its tensor of elastic constants. These elastic constants are a measurement of the material's interatomic forces, i.e., they are defined as second derivatives of the free energy with respect to strain in different symmetries, which makes them a useful tool to study the atomic environment of a crystal lattice. Experimental techniques such as resonant ultrasound spectroscopy (RUS) have proven to be very valuable in the determination of the elastic moduli of a material. RUS uses the resonance modes of elastic bodies to infer different material properties such as their elastic moduli. This experimental technique is divided into two different procedures known as the forward problem, which deals with the determination of the resonance modes from the elastic moduli of the body being studied, and the inverse problem which determines the elastic moduli from the experimental measurements of the resonance modes. In this project, a new approach to the inverse problem in RUS is presented through the use of regression trees in the context of machine learning. The results obtained show that the implementation of machine learning strategies in RUS can reduce the variance of the results achieved (when compared to traditional solutions) and can eliminate the need to have initial guesses for the elastic moduli when solving the inverse problem.

Contents

List of Figures

List of Tables

1 Introduction

The elastic reaction of a material is dictated by its tensor of elastic constants. These elastic constants are a measurement of the material's interatomic forces, i.e., they are defined as second derivatives of the free energy with respect to strain in different symmetries, which makes them a useful tool to study the atomic environment of a crystal lattice. The study of the elastic constants of a material is involved in many different areas regarding solid-state physics. For instance, these constants are important parameters when studying equations of state, lattice dynamics and phonon spectra, but also to study electronic degrees of freedom in a material, as the cohesion energies of atoms depend strongly on their constituent electrons. In addition, the elastic constants are heavily linked to quantities in thermodynamics such as the thermal coefficient and in phase transitions that can result in novel electronic ground states, such as superconductivity, charge density waves, magnetism, topological states, among others. This is why the study of the elastic constants of a material is interesting, not only to engineers, but to researchers in different areas of fundamental and applied physics. [1]

Measuring the elastic constants of a material is not a straightforward procedure since different theoretical calculations and experimental methods must be applied in order to obtain the values associated to these. Nowadays, there are numerous techniques that can successfully measure the elastic tensor of a solid. However, the accuracy of a given experimental method usually depends on many variables and parameters that must be efficiently controlled to avoid incurring in experimental errors. One way to eliminate some of these variables is through the measurement of the normal modes of a solid since these resonance frequencies are directly related to its elastic tensor. Therefore, experimental techniques such as resonant ultrasound spectroscopy have proven to be very valuable in the determination of the elastic moduli of a material. [1]

1.1 Resonant Ultrasound Spectroscopy

Resonant ultrasound spectroscopy (RUS) is an experimental technique that uses the resonance modes of elastic bodies to infer different material properties such as their elastic moduli. Through analytical and computational tools, the complete elastic tensor of an arbitrary body can be inferred from a single measurement. In order to do this, RUS is divided into two different procedures known as the forward problem and the inverse problem. The forward problem deals with the determination of the resonance modes from the elastic moduli of the body being studied, while the inverse problem determines the elastic moduli from the experimental measurements of the resonance modes [2]. In references [1, 2, 3], a complete description of this experimental

technique can be found.

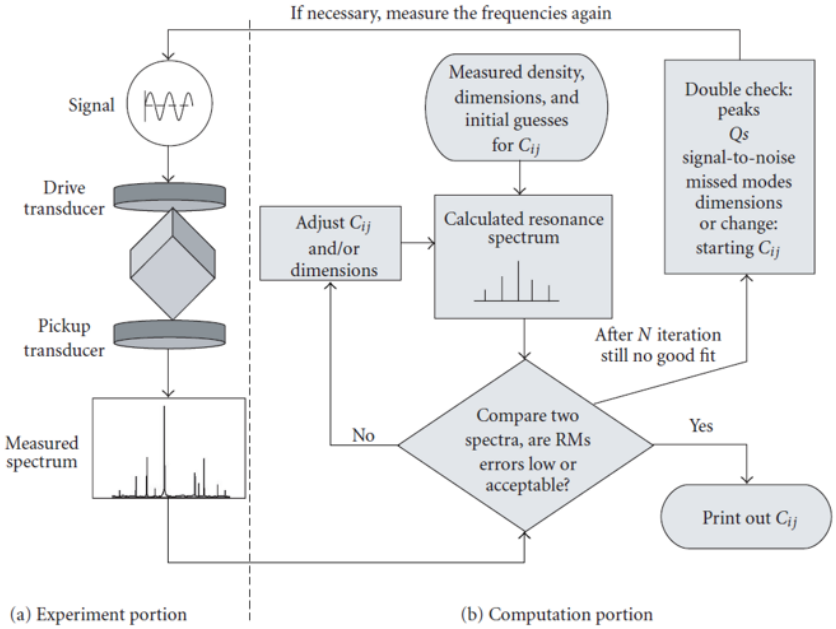

Figure 1: Diagram of the experimental and computational procedures involved in RUS. [1]

1.1.1 Forward Problem

The forward problem, regarding the determination of the resonance frequencies from the elastic moduli is a straightforward procedure. Knowing the elastic moduli, the dimensions and the density of the specimen that is being studied, the resonance frequencies can be determined through the minimization of the stationary lagrangian. This method is based on the Hamilton's principle of least action which dictates that the lagrangian of a defined system is invariant with respect to small perturbations. A complete derivation of the resonance frequencies from the elastic moduli can be found in section 1.2 and in references [1, 2, 4].

1.1.2 Inverse Problem

Experimentally, resonant ultrasound spectroscopy directly measures the resonance frequency spectrum, meaning that the elastic moduli need to be established from these frequencies. Since there is no analytical way to solve this problem, an objective function is defined using the difference of squares between the measured frequencies

and the computed frequencies (which are computed from an initial guess of the elastic moduli and the forward problem). This objective function, generally the square of the error:

$$g = \sum_i \left(f_i^{(c)} - f_i^{(m)} \right)^2 \tag{1}$$

where $f_i^{(c)}$ are the computed frequencies and $f_i^{(m)}$ are the measured frequencies, is then minimized through an iterative process and the elastic moduli are determined from this minimization. Due to the non-linearity of this problem, the Levenberg-Marquardt algorithm has often been used to solve this minimization [4]. This algorithm is a combination between the Gauss-Newton algorithm and the gradient descent. Fundamentally, the Levenberg-Marquardt algorithm varies between Gauss-Newton and the gradient descent in an adaptive manner, to update the parameters of the model (i.e the elastic moduli) and converge to a minimum of the objective function [5].

1.2 Deep Learning

Deep learning is a machine learning algorithm that takes and input and uses it to predict an output. So, given a sufficiently large data set of input and output pairs, a deep learning algorithm will minimize the difference between its predictions and the real (or expected) outputs. This is done by learning or associating different patterns between the given pairs of input and output leading the deep learning model to generalize these patterns to inputs that it has not seen before [6].

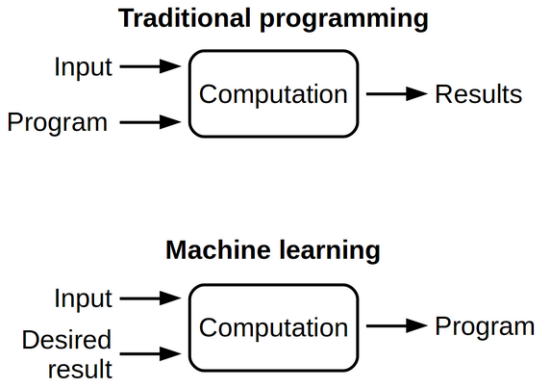

Figure 2: Difference between traditional programming and machine learning. [7]

The key differences between traditional programming and machine learning can be seen in figure 2. The starting points for traditional programming and machine learning are similar since both aim to solve problems regarding inputs and outputs, but the differences are in the execution. Instead of trying to write down a program to solve a specific problem, in machine learning, input data and desired target values are collected to instruct a computer to find a program that computes an output for each given input value [7]. To do this, there are several algorithms such as tree-based models and neural networks which can find such program. More details on these specific algorithms can be found in section 2.

Considering this, the purpose of this work is to implement and evaluate different deep learning models to find an alternative to the traditional solutions used to solve the inverse problem in resonant ultrasound spectroscopy.

2 Background theory

2.1 Obtaining the resonance frequencies from the elasticity tensor

Starting with classical mechanics, it is known that, in absence of external loads, the general Lagrangian of an arbitrary body of volume V is given by [8]:

$$\hat{\mathcal{L}} = \int_V (K - U) \ \mathrm{d}V \tag{2}$$

Where K is the kinetic energy of such volume and U refers to its potential energy. Now, considering a linear elastic body of an arbitrary shape, its kinetic energy is given by [8]:

$$K = \frac{\rho}{2} \frac{\partial \hat{u}_i}{\partial t} \frac{\partial \hat{u}_i}{\partial t} \tag{3}$$

In this case, ρ is the density of the body and $\hat{u}_i$ is the unit vector of the displacement of the body. In order to simplify the following equations, Einstein's notation will be used throughout the rest of this mathematical development. This means that repeated indices within the same term of an equation represent a summation over the range of such index. Thus, considering the same linear elastic body, the potential energy is given by the strain energy [8]:

$$U = \frac{1}{2} C_{ijkl} \varepsilon_{ij} \varepsilon_{kl} \tag{4}$$

The indices i, j, k, l have a range from 1 to 3, representing the three different directions in which strain can be present. Within this equation, Hooke's Law can be identified, where [8]:

$$\sigma_{ij} = C_{ijkl} \varepsilon_{kl} \tag{5}$$

In this equation, σ_{ij} is known as the stress tensor, C_{ijkl} as the elasticity tensor and ε_{kl} as the strain tensor. Thus [8]:

$$U = \frac{1}{2} \sigma_{ij} \varepsilon_{ij} \tag{6}$$

It is important to mention that in the absence of external loads, no torque is present in the elastic body being studied, meaning that the stress tensor is symmetric. Now, assuming that the dependence of the displacement with respect to time is harmonic, the displacement can be written as [8]:

$$\hat{u}_i\left(\vec{x},t\right) = u_i(\vec{x})e^{i\omega t} \tag{7}$$

And, taking into account that the strain tensor for a linear elastic body is given by [8]:

$$\varepsilon_{kl} = \frac{1}{2}\left(\hat{u}_{k,l} + \hat{u}_{l,k}\right) = \frac{1}{2}\left(u_{k,l} + u_{l,k}\right)e^{i\omega t} \tag{8}$$

Where $u_{k,l}$ represents $\partial u_k/\partial x_l$. Therefore, considering that both the stress and strain tensors are symmetric, the elasticity tensor is symmetric in its first two indices and its last two indices. These are called "minor symmetries", and the "major symmetry" i.e. $C_{ijkl} = C_{klij}$ comes from energy considerations. Taking into account that all three tensors are symmetric, the Voigt notation can be used. This notation introduces the following contractions regarding two indices [9]:

$$\begin{array}{ccc} 11 \to 1 & 22 \to 2 & 33 \to 3 \\ 23 = 32 \to 4 & 13 = 31 \to 5 & 12 = 21 \to 6 \end{array} \tag{9}$$

This notation simplifies the elasticity tensor, converting it from a 3x3x3x3 tensor to a 6x6 tensor and both the stress and strain tensor convert from a 3x3 tensor to a 6x1 tensor. With this, Hooke's Law can be rewritten as [8]:

$$[\sigma] = [C][\varepsilon] \tag{10}$$

Where:

$$[\sigma] = [\sigma_1\ \sigma_2\ \sigma_3\ \sigma_4\ \sigma_5\ \sigma_6]^T \tag{11}$$

$$[\varepsilon] = \begin{bmatrix} u_{1,1} \\ u_{2,2} \\ u_{3,3} \\ \frac{1}{2}\left(u_{2,3} + u_{3,2}\right) \\ \frac{1}{2}\left(u_{1,3} + u_{3,1}\right) \\ \frac{1}{2}\left(u_{1,2} + u_{2,1}\right) \end{bmatrix} e^{i\omega t} \tag{12}$$

And:

$$[C] = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \tag{13}$$

Therefore, the potential energy of the body being studied can be expressed in matrix notation as:

$$U = \frac{1}{2}[\sigma]^T[\varepsilon] = \frac{1}{2}[\varepsilon]^T[C]^T[\varepsilon] = \frac{1}{2}[\varepsilon]^T[C][\varepsilon] \tag{14}$$

Now, considering the kinetic energy and remembering the assumption that the displacements are harmonic with respect to time, the magnitude of this energy can be written as:

$$K = \frac{\rho\omega^2}{2}[u]^T[u]e^{2i\omega t} \tag{15}$$

Where:

$$[\hat{u}] = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} e^{i\omega t} = [u]e^{i\omega t} \tag{16}$$

Finally, in order to express the Lagrangian in terms of the displacement, the strain tensor can be express in terms of the displacement and the symmetric gradient tensor,

such that [8]:

$$[\varepsilon] = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} \\ 0 & \frac{1}{2}\frac{\partial}{\partial x_3} & \frac{1}{2}\frac{\partial}{\partial x_2} \\ \frac{1}{2}\frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_1} \\ \frac{1}{2}\frac{\partial}{\partial x_2} & \frac{1}{2}\frac{\partial}{\partial x_1} & 0 \end{bmatrix} [u]e^{i\omega t} = [B][u]e^{i\omega t} \tag{17}$$

Thus, the potential energy can be rewritten as:

$$U = \frac{1}{2}[u]^T[B]^T[C][B][u]e^{2i\omega t} \tag{18}$$

Having completed this, the Lagrangian defined at the beginning of this procedure can be expressed as:

$$\hat{\mathcal{L}} = \frac{1}{2}\int_V \left(\rho\omega^2[u]^T[u] - [u]^T[B]^T[C][B][u]\right) e^{2i\omega t}\ \mathrm{d}V \tag{19}$$

Now, since the right side of this equation is harmonic, therefore the left side must also be harmonic [8]. This allows the Lagrangian to be expressed as:

$$\hat{\mathcal{L}} = \mathcal{L}e^{2i\omega t} \tag{20}$$

Where:

$$\mathcal{L} = \frac{1}{2}\int_V \left(\rho\omega^2[u]^T[u] - [u]^T[B]^T[C][B][u]\right)\ \mathrm{d}V \tag{21}$$

Having obtained an expression for the Lagrangian of the problem, it is necessary to find a way to express the displacements such that they can be evaluated numerically. The easiest way to do this, in terms of accuracy and computation time, is by expanding each displacement term in a Cartesian power series of the form [8]:

$$[\Psi_\lambda] = [x_1^p x_2^q x_3^r] \tag{22}$$

Once again, x_1, x_2, x_3 represent the three different directions and $p, q, r \in N$.

Considering this power series as a basis, each component of the displacement tensor can be written as:

$$u_i = \sum_{(p,q,r)}^{\infty} a_{i(p,q,r)} \Psi_{(p,q,r)} = a_{i\lambda} \Psi_\lambda \tag{23}$$

Where $a_{i\lambda}$ is a constant. Clearly, (p, q, r) cannot be summed to infinity when using computational tools. Therefore, a truncation of the sum is used such that $p + q + r \leq N$ where N is the truncation number. It is important to mention that as N tends to infinity, the linear combination of the elements in the basis tend to the actual displacements. Thus N is defined as the lowest number that will allow a favorable compromise between accuracy and computation time [8]. The number of terms in the polynomial expansion is given by:

$$\binom{N+3}{3} = \frac{(N+3)!}{3!N!} = \frac{1}{6}\frac{(N+3)!}{N!}$$

With the use of this basis, the displacements can be written as a product of a square tensor containing vectors composed of monomials and a column vector containing the respective constants, such that:

$$[u] = \begin{bmatrix} [\phi] & [0] & [0] \\ [0] & [\phi] & [0] \\ [0] & [0] & [\phi] \end{bmatrix} \begin{bmatrix} [a_1] \\ [a_2] \\ [a_3] \end{bmatrix} = [\Phi][a] \tag{24}$$

Each $[\phi]$ depends on the number chosen for N. For example, if $N = 2$:

$$[\phi] = [1 \;\; x_1 \;\; x_2 \;\; x_3 \;\; x_1x_2 \;\; x_1x_3 \;\; x_2x_3 \;\; x_1^2 \;\; x_2^2 \;\; x_3^2] \tag{25}$$

Then, substituting this in the latter expression found for the lagrangian:

$$\mathcal{L} = \frac{1}{2}\int_V \left(\rho\omega^2 [a]^T [\Phi]^T [\Phi][a]\right) \, \mathrm{d}V - \frac{1}{2}\int_V \left([a]^T [\Phi]^T [B]^T [C][B][\Phi][a]\right) \, \mathrm{d}V \tag{26}$$

In order to write this equation in a more compact way, one can define the following integrals as matrices (considering that $[a]$ are simply constants) [8]:

$$[\mathcal{E}] = \int_V [\Phi]^T \rho [\Phi] \, \mathrm{d}V \tag{27}$$

$$[\Gamma] = \int_V [\Phi]^T [B]^T [C][B][\Phi] \, \mathrm{d}V \tag{28}$$

It is important to mention that both matrices $[\mathcal{E}]$ and $[\Gamma]$ are symmetric and square, and $[\mathcal{E}]$ is positive definite as well [8]. Thus:

$$\mathcal{L} = \frac{1}{2}\omega^2 [a]^T [\mathcal{E}][a] - \frac{1}{2}[a]^T [\Gamma][a] \tag{29}$$

Having found the correct way to express the Lagrangian of the body being studied, the variations of this Lagrangian can be calculated. In the equilibrium state, the Lagrangian must be stationary with respect to any variation, that is $\mathrm{d}\mathcal{L} = 0$ [8]. The variation of the Lagrangian is given by:

$$\begin{aligned} \mathrm{d}\mathcal{L} = & \frac{\partial \mathcal{L}}{\partial a_{10}} \mathrm{d}a_{10} + ... + \frac{\partial \mathcal{L}}{\partial a_{1(R/3)}} \mathrm{d}a_{1(R/3)} \\ & + \frac{\partial \mathcal{L}}{\partial a_{20}} \mathrm{d}a_{20} + ... + \frac{\partial \mathcal{L}}{\partial a_{2(R/3)}} \mathrm{d}a_{1(R/3)} \\ & + \frac{\partial \mathcal{L}}{\partial a_{30}} \mathrm{d}a_{30} + ... + \frac{\partial \mathcal{L}}{\partial a_{3(R/3)}} \mathrm{d}a_{3(R/3)} \end{aligned} \tag{30}$$

Where R is the number of columns in the matrix $[\Phi]$, such that:

$$R = 3 \binom{N+3}{3} = 3\frac{(N+3)!}{3!N!} = \frac{1}{2}\frac{(N+3)!}{N!} \tag{31}$$

Now, since each coefficient in the variation of the Lagrangian is free to vary arbitrarily, $\mathrm{d}\mathcal{L} = 0$ if each term is independently 0. This can be written as:

$$\frac{\partial \mathcal{L}}{\partial [a]} = \omega^2 [a]^T [\mathcal{E}] - [a]^T [\Gamma] = [0]^T \tag{32}$$

Arriving to:

$$\omega^2[a]^T[\mathcal{E}] = [a]^T[\Gamma] \tag{33}$$

Finally, considering that both $[\mathcal{E}]$ and $[\Gamma]$ are symmetric, then this equation can be rewritten as:

$$\boxed{\omega^2[\mathcal{E}][a] = [\Gamma][a]} \tag{34}$$

Which is the generalized eigenvalue problem that is found in the literature regarding the forward problem in Resonant Ultrasound Spectroscopy [1]. In this case, ω^2 are the eigenvalues of the problem and $[a]$ are the respective eigenvectors. Therefore, the forward problem in RUS is reduced to calculating the volume integrals defined as $[\mathcal{E}]$ and $[\Gamma]$ [8].

These integrals can be evaluated analytically for different shapes such as rectangular parallelepipeds, cylinders, spheroids, and ellipsoids [8]. In the specific case of a rectangular parallelepiped of dimensions 2α, 2β and 2γ, the volume integrals take the form:

$$\mathcal{D} = \int_{-\alpha}^{\alpha}\int_{-\beta}^{\beta}\int_{-\gamma}^{\gamma} x_1^p x_2^q x_3^r \, \mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3 \tag{35}$$

Which has the analytical solution of:

$$\mathcal{D} = \frac{8\alpha^{p+1}\beta^{q+1}\gamma^{r+1}}{(p+1)(q+1)(r+1)} \tag{36}$$

And similarly, the same integrals for the cylinders, spheroids and ellipsoids can be carried out. Last but not least, the solution to equation 34 has an infinite set of eigenvalues, therefore, as it was mentioned before, a truncation of the power series presented in equation 22 has to be performed. Thus, N is usually varied until the first 50 modes appear to converge. The standard value that has been used for simple geometries varies from 10 to 20, making the rank of the matrix $[\Phi]$ range from 858 to 4620. But the number of modes needed to determine the parameters of the material being studied, depends on the symmetry of the material. Meaning that for anisotropic materials, there may be up to 21 independent elastic constants, while for an isotropic material there are only 2 independent elastic constants. Figure 3 shows the independent elastic moduli for each type of crystal symmetry.

Crystal class	Number of C_{ij}	List of elastic constants
Triclinic	21	All possible combinations
Monoclinic	13	C_{11}; C_{12}; C_{13}; C_{16}; C_{22}; C_{23};C_{26}; C_{33}; C_{36}; C_{44}; C_{45}; C_{55}; C_{66}
Orthorhombic	9	C_{11}; C_{12}; C_{13}; C_{22}; C_{23}; C_{33}; C_{44}; C_{55}; C_{66}
Trigonal	6 or 7	C_{11}; C_{33}; C_{44}; C_{13}; C_{12}; C_{14}; C_{25}
Tetragonal	6	C_{11}; C_{33}; C_{44}; C_{13}; C_{12}; C_{66}
Hexagonal	5	C_{11}; C_{33}; C_{44}; C_{12}; C_{14}
Cubic	3	C_{11}; C_{12}; C_{44}
Isotropic	2	C_{11}; C_{44}

Figure 3: Independent elastic moduli for each type of crystal symmetry [1].

Based on a consensus of experienced RUS practitioners, it has been determined that the number of modes needed to accurately fit the elastic constants of a material being studied should be between 8 to 10 times the number of independent elastic constants [8].

2.2 Obtaining the elasticity tensor from the resonance frequencies

As it was mentioned in the introduction, there is no analytical way to obtain the elasticity tensor from the resonance frequencies. Thus, numerical methods are used in order to solve this inverse problem. Commonly, two different methods have been used when trying to solve the inverse problem in RUS, the Levenberg-Marquadt Algorithm, and the Genetic Algorithm [10].

2.2.1 Levenberg-Marquadt Algorithm

The use of this algorithm starts by introducing an objective function named as χ^2 defined by using the difference of squares between the measured frequencies ($[f^{(m)}]$) and the computed frequencies ($[f^{(c)}]$) which are obtained from an initial guess of the elastic moduli considered as the parameters of the model ($[p]$), and the forward problem. In the specific case of RUS, χ^2 is defined as:

$$\chi^2 = \sum_i \left(\frac{f_i^{(m)} - f_i^{(c)}}{\sigma_i} \right)^2 \tag{37}$$

$$\chi^2 = \left([f^{(m)}] - [f^{(c)}] \right)^T [Q] \left([f^{(m)}] - [f^{(c)}] \right) \tag{38}$$

Where $[Q]$ is a diagonal matrix such that $Q_{ii} = 1/\sigma_i^2$ and σ_i refers to the error associated with $f_i^{(m)}$. Expanding this product of matrices, the final equation for the objective function is found [11]:

$$\chi^2 = [f^{(m)}]^T[Q][f^{(m)}] - 2[f^{(m)}]^T[Q][f^{(c)}] + [f^{(c)}]^T[Q][f^{(c)}] \tag{39}$$

Thus, the goal of the Levenberg-Marquadt Algorithm is to minimize the objective function in order to find the parameters of the theoretical model that fit the experimental data accurately [5]. To do this, the parameters of the model need to be concurrently updated such that [11]:

$$\chi^2([p] + [h]) < \chi^2([p])$$

Until a minimum is reached ($[h]$ refers to the parameter update). Traditionally the parameter update can be found by using the Gauss-Newton Algorithm or the Gradient Descent Algorithm [11]. Starting with the Gauss-Newton Algorithm, the parameter update ($[h_{gn}]$) is found by setting:

$$\frac{\partial \chi^2([p] + [h_{gn}])}{\partial [h_{gn}]} = 0 \tag{40}$$

Through a first-order Taylor approximation [5]:

$$[f^{(c)}]([p] + [h_{gn}]) \approx [f^{(c)}]([p]) + \left[\frac{\partial [f^{(c)}]}{\partial [p]} \right] [h_{gn}] \approx [f^{(c)}]([p]) + [P][h_{gn}] \tag{41}$$

Thus, calculating χ^2 in the updated parameters yields to:

$$\begin{aligned}\chi^2([p]+[h_{gn}]) =&[f^{(m)}]^T[Q][f^{(m)}]+[f^{(c)}]^T[Q][f^{(c)}]\\ &-2[f^{(m)}]^T[Q][f^{(c)}]\\ &-2\left([f^{(m)}]-[f^{(c)}]\right)[Q][P][h_{gn}]\\ &+[h_{gn}]^T[P]^T[Q][P][h_{gn}]\end{aligned} \tag{42}$$

Taking the derivative with respect to $[h_{gn}]$:

$$\frac{\partial\chi^2\left([p]+[h_{gn}]\right)}{\partial[h_{gn}]} \approx -2\left([f^{(m)}]-[f^{(c)}]\right)[Q][P]+2[h_{gn}]^T[P]^T[Q][P] \tag{43}$$

Therefore, in order to satisfy equation 40:

$$\left([P]^T[Q][P]\right)[h_{gn}] = [P]^T[Q]\left([f^{(c)}]-[f^{(m)}]\right) \tag{44}$$

This equation is referred to as the Gauss-Newton update equation [11]. Now, considering the Gradient Descent, the parameter update is found by finding the direction of the steepest descent in the context of the problem [11]. To do this, the derivative of the objective function with respect to the parameters is found, where:

$$\frac{\partial\chi^2}{\partial[p]} = 2\left([f^{(m)}]-[f^{(c)}]\right)^T[Q]\frac{\partial\left([f^{(m)}]-[f^{(c)}]\right)}{\partial[p]} \tag{45}$$

$$\frac{\partial\chi^2}{\partial[p]} = -2\left([f^{(m)}]-[f^{(c)}]\right)^T[Q]\frac{\partial[f^{(c)}]}{\partial[p]} = -2\left([f^{(m)}]-[f^{(c)}]\right)^T[Q][P] \tag{46}$$

Therefore, the parameter update for the Gradient Descent Algorithm is given by:

$$[h_{gd}] = \alpha\left([f^{(m)}]-[f^{(c)}]\right)^T[Q][P] \tag{47}$$

Where α is simply the length of the step. Now, going back to Levenberg-Marquardt, this algorithm combines both Gauss Newton and Gradient Descent by introducing a damping parameter μ, such that the parameter update (h_{lm}) is given by:

$$\boxed{\left([P]^T[Q][P]+\mu[I]\right)[h_{lm}] = [P]^T[Q]\left([f^{(c)}]-[f^{(m)}]\right)} \tag{48}$$

Small values of μ lead to the Gauss-Newton algorithm, while large values of μ result in the gradient descent algorithm (this can be seen better in figure 4). The damping parameter μ is usually initialized as a large value in order to have small steps in the steepest-descent direction at the beginning of the iterations. If $\chi^2([p]+[h]) > \chi^2([p])$, μ is increased. Otherwise, μ is decreased and the method approaches to the Gauss-Newton algorithm, resulting in a minimum of χ^2 [5]. This makes the Levenberg-Marquardt algorithm somewhat robust to poor initial guesses. The program used for the solution of this problem can be found in [4].

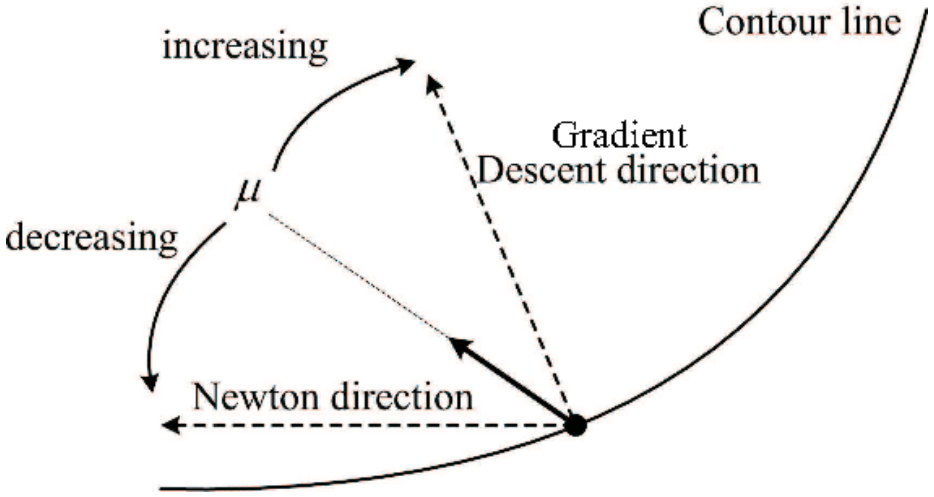

Figure 4: Levenberg-Marquadt algorithm. [12]

2.2.2 Genetic Algorithm

The method described before relies on the assumption that there are no missing resonances in the experimental dataset, which is not always valid because some modes may not be excited for a given sample transducer geometry. To solve this problem, the Genetic Algorithm has been used in RUS. This algorithm is both robust to poor initial guesses and to missing frequencies, but it is very costly when considering computation time. To use this type of algorithm, the following steps are used [10]:

(i) N different sets of arbitrary elastic moduli are generated, where N is approximately 10 times the number of constants within each set. These sets are denoted as the parent moduli $x_i^{\mathbf{parent}}$.

(ii) N mutant sets of elastic moduli are generated by using the parent sets such that: $x_i^{\mathbf{mutant}} = x_j^{\mathbf{parent}} + \lambda(x_k^{\mathbf{parent}} - x_l^{\mathbf{parent}})$. In this case, j, k, l are random integers ranging from 1 to N and λ is an scaling factor which is set by the user.

Usually λ take small values such that the mutant sets are small variations from the parent sets.

(iii) The child sets $x_i^{\mathbf{child}}$ are created by performing a crossing operation between the parent sets and the mutant sets. Each child set is created by taking the parent set with a probability of p or the mutant set with a probability of $(1-p)$. This step must be repeated individually for each modulus within each set.

(iv) The resonance frequencies for the parent sets and the child sets are calculated through the forward problem presented before. Then, the sum-of-squares residuals between the measured and computed frequencies is calculated, and the set with the lowest residual (when comparing parent set vs. child set) is chosen for the next generation.

(v) A new parent set is selected from the last step and the procedure is repeated until a satisfactory convergence is achieved.

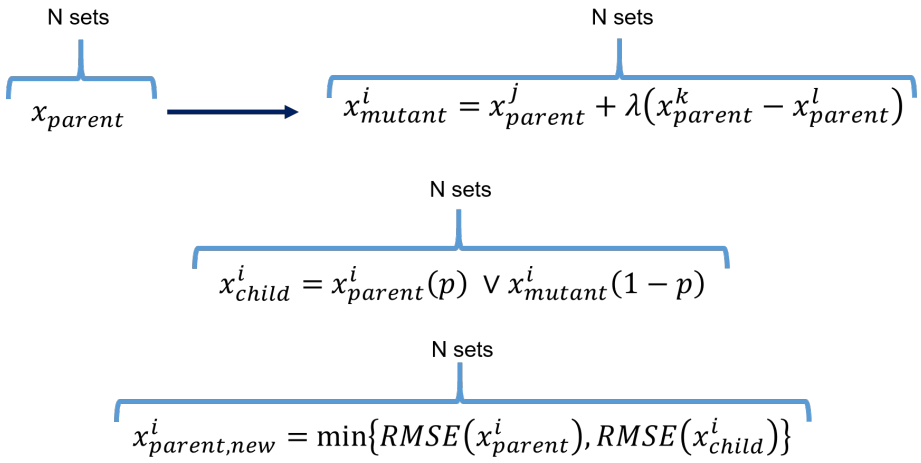

Figure 5: Process involved in the genetic algorithm in the case of no missing frequencies.

In the case that it is suspected that not all resonance frequencies have been detected experimentally, there is a slight change in the step (iv) of this algorithm. Within this same step, the following substeps must be added [10]:

(i) The user has to input the maximum number of frequencies that they supposed are missing from the experimental data.

(ii) The residuals are calculated with the experimental data disregarding the missing frequencies. Then, one missing frequency is inserted between the first and the second frequency in the experimental data and the residuals are calculated again. The inserted frequency has a weight of zero considering the residuals, since the exact value of the missing frequency is unknown, but it still shifts

the higher frequencies by one place. The missing frequency is then inserted between the second and third frequency, then between the third and fourth and so on. The process is then repeated with all the possible combinations of two missing frequencies, then three and so on (until reaching the number of missing frequencies given by the user). Considering this, the combination with the lowest residuals is taken for the comparison between the parent and the child set.

(iii) At the end of each generation, the user can see where the missing frequencies are, and where to find them in the experimental range.

$$x^i_{parent,new\,0} = \min\{RMSE(x^i_{parent}), RMSE(x^i_{child})\}$$

$$m = number\ of\ missing\ frequencies$$

$$x^i_{parent,new\,1} = \min\left\{RMSE\left(x^i_{parent}(1\ freq)\right), RMSE\left(x^i_{child}(1\ freq)\right)\right\}$$

$$\downarrow$$

$$x^i_{parent,new\,m} = \min\left\{RMSE\left(x^i_{parent}(m\ freq)\right), RMSE\left(x^i_{child}(m\ freq)\right)\right\}$$

N sets

$$x^i_{parent,new} = \min\{RMSE(x^i_{parent,new\,0}), \dots, RMSE(x^i_{parent,new\,m})\}$$

Figure 6: Alteration of step (iv) in the genetic algorithm to include possible missing frequencies.

Clearly this is a straightforward procedure, but it scales very quickly when the number of missing frequencies increases. For example, if in the experimental data there are 60 identified frequencies, but the user predicts that there are 5 missing frequencies, the residuals must be calculated 5,461,512 times per generation. And if there are 15 missing frequencies, the residuals must be calculated over 53 trillion times per generation [10].

Finally, with either of these two algorithms the logarithmic derivatives of each resonance frequency with respect to each modulus are computed numerically such that [10]:

$$\alpha_{i,\mu} = \frac{\partial f_\mu}{\partial C_i} \cdot \frac{C_i^0}{f_\mu^0} \tag{49}$$

Where f_μ is each of the resonance frequencies, C_i is each elastic modulus, and

f_μ^0 and C_i^0 are the resonance frequencies and elastic moduli respectively at a reference temperature of 300 K. To a first approximation, these α factors are essentially geometric, meaning that they can be considered independent of the temperature at which measurements are being done. If this is the case, the following equation can be used to extract the temperature dependence of elastic constants (the useful thermodynamic quantities) with the temperature dependence of the resonance frequencies (which are measured in RUS) [10]:

$$\frac{2\Delta f_\mu(T)}{f_\mu^0} = \sum_i \alpha_{i,\mu} \cdot \frac{\Delta C_i(T)}{C_i^0} \tag{50}$$

Meaning that these factors present another way for determining the dependence of the elastic moduli with respect to the temperature. Lastly, these factors are normalized such that [10]:

$$\sum_i \alpha_i = 1 \tag{51}$$

2.3 Deep Regression

As it has been shown in the last sections, the inverse problem in Resonant Ultrasound Spectroscopy is mainly concerned as a regression problem. Regression techniques have been widely used to solve tasks where the final goal is to predict continuous values [13]. In this specific context, the goal is to predict the elastic moduli from the resonance frequencies. Within the last decade, deep learning architectures have shown to outperform the state-of-the-art solutions in this regard [13]. Therefore, the most commonly used deep regression techniques are discussed in the following sections.

2.3.1 Regression Trees

Decision trees dependent on If–Then principles are perhaps the most well-known strategies utilized in machine learning for classification since they offer outputs that can be effectively understood. Accordingly, this method obtains ordered arrangements of If–Then guidelines for forecasting, which produces justifiable models. Considering this, decision trees have also been used as regression techniques. Known as regression trees, these types of algorithms consequently offer the power of a non-parametric regression procedure, which is regularly better compared to common least squares assessments when parametric conditions are not met. As well as this, regression trees offer an open-box model which is frequently needed by users with low expertise in

machine learning. [14]

It is important to mention that these specific regression techniques have been classified as **off-the-shelf** regressors. This means that these models can be applied directly to different contexts of data without requiring a great deal of time consuming data pre-processing or careful tuning of the learning procedure [15]. There are several tree-based regressors which can prove to be useful in the specific context that is being studied. The most common algorithm is known as random forest regressor. A random forest regressor is an aggregation of decision trees that consist of a series of splitting rules, starting at the top of the tree. The following example taken from [16] is useful to understand how a decision tree works. The tree presented in 7 is an attempt to predict a baseball player's salary based on the number of years that he has played and the number of hits that he made in the previous year [16].

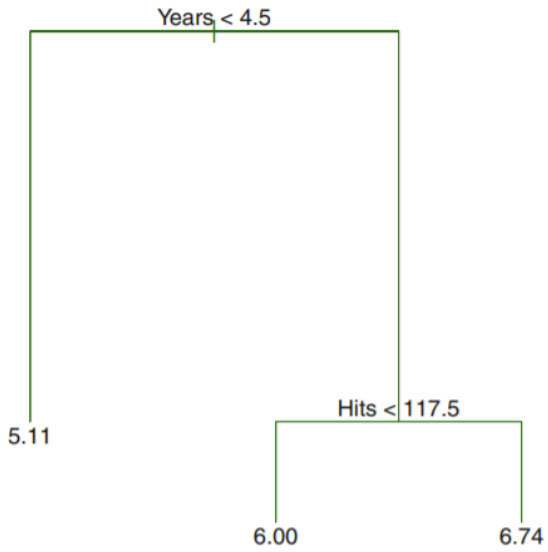

Figure 7: Decision tree used to predict a baseball player's salary [16].

From this tree the number of years is the most important factor in determining salary, and players with less experience earn lower salaries than more experienced players. But among players who have been playing for five or more years, the number of hits made affects the salary, and players who made more hits last year tend to have higher salaries. This is a basic example of how a decision tree works on the context of regression, in reality, decision trees are much more complex containing a higher number of splitting rules. So, when using a random forest regressor, a number N of trees (denominated as the forest) are fitted, and the final result is the average of the whole forest. [16]

2.3.2 Neural Networks

Deep learning algorithms usually use neural networks in order to find such associations with the inputs and outputs. In figure 8, the classical structure of a neural network can be seen. A neural network is made up of an input layer, certain hidden layers, and an output layer. These layers are composed of nodes which can be identified as the white circles in figure 8. In practice, the input layer takes in numerical values, while the hidden layers search for patters or associations and the output layer gives the predictions. [6]

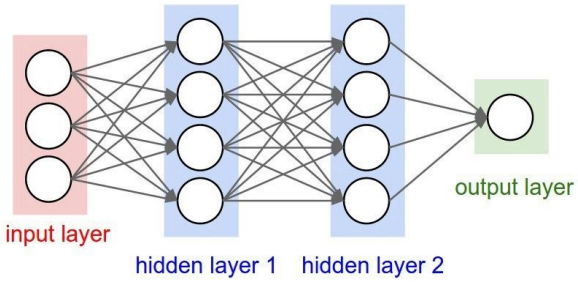

Figure 8: Basic neural network architecture. [6]

Each node is connected to other nodes with what is called a *weight* and a *bias*. These *weights* and *biases* are tuned by the neural network in order to minimize the difference between the predicted outputs and the expected outputs [6]. Therefore, the goal of a neural network is to minimize the difference between the actual outputs and the predicted ones, and for this, *back-propagation* is used. In general terms, *back-propagation* is a gradient descent where the network backtracks through its layers to update both the weights and biases to minimize the value of the loss function. [6]

Considering this, deep learning appears when deep neural networks are created. This happens when a neural network has a considerable number of layers. Thus, the complexity of the architecture is higher, and more *weights* and *biases* are present, which can improve the results given by the network [6]. In deep learning, convolutional neural networks are commonly used. This architecture is used to perform complex tasks regarding images, sounds or even texts. Its main feature is the convolution layer, since it applies a filter or kernel to an input image which gets a feature map that highlights characteristics of the image in order to simplify the classification and recognition process [6]. In the following figure, the architecture of an arbitrary convolutional neural network can be seen.

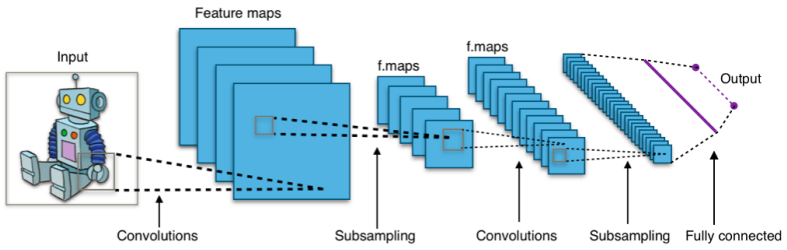

Figure 9: Common convolutional neural network architecture. [6]

So, as it can be seen, the convolutional layers are used multiple times in within the network in order to refine the feature maps. The refinement of the feature maps usually results in better predictions and lower convergence times [6]. In 2019, Heshemi [17] proposed a way to train neural networks without the need of labeled data by simply inserting the physics and boundary conditions of the problem being analyzed. Based on the U-net architecture with custom physics-informed loss functions, an accuracy higher than 99.9% was obtained for three different problems involving the solution of partial differential equations, even nonlinear and non-homogenous partial differential equations [17].

In this case, the use of neural networks as a specific regression technique has been classified as a **non-off-the-shelf** regressor. Contrary to the off-the-shelf regressors, this means that these models cannot be applied directly to different contexts of data without requiring a great deal of time consuming data pre-processing or careful tuning of the learning procedure. Thus, a thorough procedure of pre-processing the data and tuning the learning process must be done in order to obtain satisfactory results. [15]

The most common libraries used to develop machine learning algorithms based on neural networks and tree-based algorithms are Scikit-Learn, TensorFlow and Pytorch. These libraries have different tools such as GPU-enabled acceleration which can prove useful to the training and optimizing sections where different iterations of different models must be done in order to find the correct hyperparameters.

2.3.3 Hyperparameter Tuning

From each deep learning strategy, there are several parameters that can be changed in order to achieve a better result. These, in the machine learning context are known as hyper-parameters, but the question relies on how to find the hyper-parameters that represent the best model. Grid search experiments, where the models are evaluated on a discrete grid of hyper-parameters have been commonly used in the literature of empirical machine learning to optimize the hyper-parameters of learning algorithms and achieve better results. Also, multistage, multi-resolution grid experiments have

been conducted because a grid experiment with a very high resolution would be absurdly expensive in terms of computation time. In 2012, Bergstra and Bengio [18] showed that random experiments are more efficient than grid experiments for hyper-parameter optimization in the case of numerous machine learning algorithms. This study suggested that random experiments are more efficient because not all hyperparameters are equally important to tune and when compared with common grid search experiments, random search found better models in most cases and required less computational time [18]. In figure 10, the difference between a grid search and a random search can be seen.

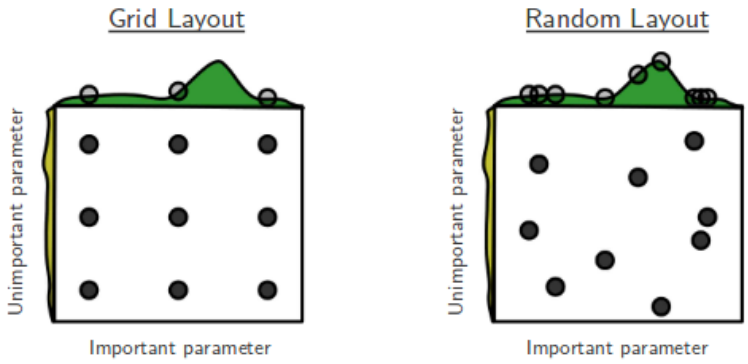

Figure 10: Difference between a grid search and a random search for hyper-parameter optimization. [18]

For this figure, a grid and random search of nine trials for optimizing a function $f(x, y) = g(x) + h(y) \approx g(x)$ with low dimensionality is presented. Above each of the two squares $g(x)$ is shown in green and in the left of each square $h(y)$ is shown in yellow. From the figure, it can be seen that $g(x)$ has a higher importance that $h(y)$ when calculating $f(x, y)$. Considering this, with grid search, nine different trials only test $g(x)$ in three distinct places. But with random search, all nine trials explore different values of g which is beneficial to the optimization of function $f(x, y)$. This malfunction when using grid search has been observed to be the rule in high dimensional hyper-parameter optimization. [18]

2.4 Related Work

In 2020, Ghosh et al. [19] trained an artificial neural network in order to learn a function that maps the jumps in resonance frequencies at a phase transition to one of two classes, corresponding to either a singlet or doublet order parameter. In figure 11 the schematic of the model used to train the neural network in this context can

be seen.

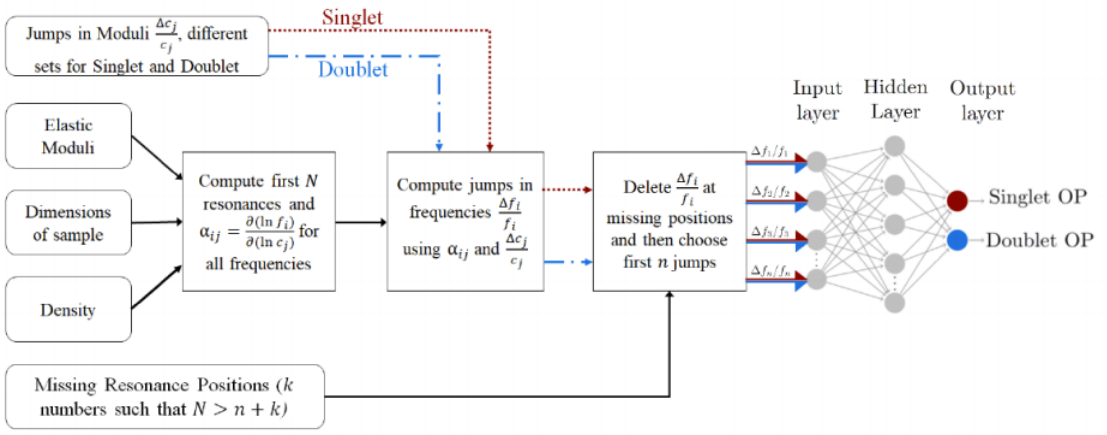

Figure 11: Schematic of the algorithm used to generate the training data in the solution presented by Ghosh et al. [19].

From the figure, it can be identified that this is a classification problem, where the outputs are only two different classes. Regardless of being related to Resonant Ultrasound Spectroscopy, the use of this neural network has a different purpose to what has been presented throughout this project. Other than this approach, no other uses of machine learning in Resonant Ultrasound Spectroscopy have been found, which gives an exciting opportunity to present a new approach to the inverse problem in RUS.

3 Methodology

3.1 Generating the data-set

To train and test the machine learning solutions presented in this thesis, the generation of the data-set was a key process. The data-set consisted of the different elastic constants, resonance frequencies and the material density, dimension and shape. Therefore, in order to obtain this data-set, the forward problem described in section 1.2 had to be implemented computationally. To do this, the implementation of the forward problem presented in the GitHub repository [20] was used. A thorough process of debugging the code presented in the repository mentioned before had to be done. As well as this, different characteristics of the code had to be changed to obtain the data needed in a more organized manner to facilitate the pre-processing stage. The resulting modified code can be found in the *Quantum Materials Research Group* RUS github repository [21].

Having successfully implemented the forward problem and having checked that the implementation was done correctly, the data-set was generated. To check the implementation of the forward problem, the resonance frequencies from known materials were calculated and compared to actual experimental results [22]. Having done this, the data-set was divided in different subsets regarding each symmetry that different materials can have (since the number of independent elastic constants depends on this symmetry). Due to time limitations, data-set was reduced to only isotropic, cubic, and tetragonal symmetries. Therefore, 150,000 elastic constants and resonance frequencies were generated for each symmetry. Specifically, since the resonance frequencies are dependent on the elastic moduli, the density, the dimensions, and the shape of the material, these were varied as follows:

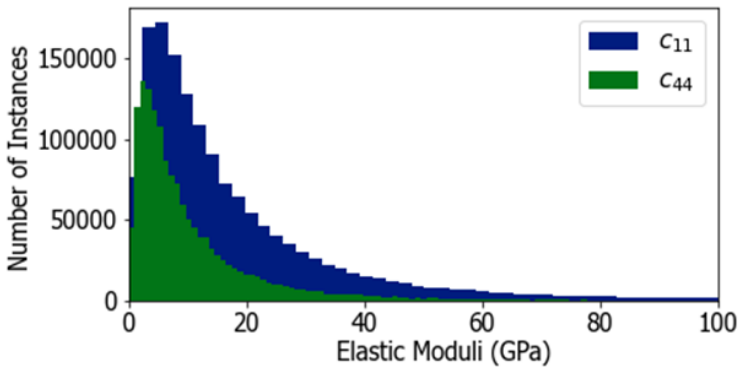

Figure 12: Distribution of elastic moduli used for the isotropic symmetry. This symmetry has two independent constants, thus c_{11} and c_{44} are shown in the figure.

- Each **elastic constant** was randomly generated using a lognormal distribution by means of the diamond constants found in [23] as the upper limits on the distribution, given that diamond presents one of the largest values of bulk modulus in nature (445 GPa) [23]. To generate each elastic constant, a stock lognormal distribution was used with mean of zero and standard deviation of one, and the distribution was multiplied by the respective elastic constant of the diamond (whether it be c_{11}, c_{44}, etc...) divided by ten.
- The **density** for each iteration was randomly generated by using a uniform distribution with the density of Styrofoam as the lower limit and the density of osmium as the upper limit, as these materials present some of the most extreme densities that can be found in solids.

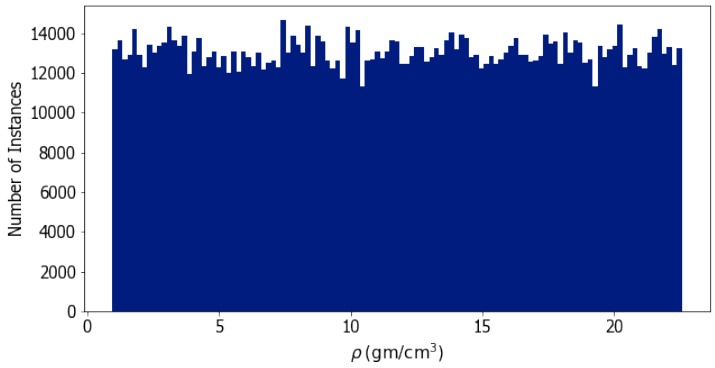

Figure 13: Distribution of density used for the isotropic symmetry.

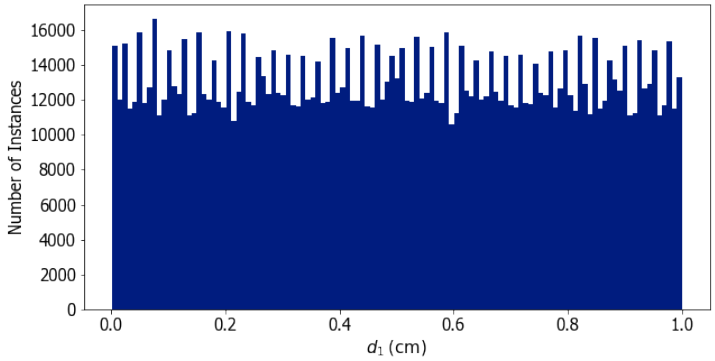

Figure 14: Distribution of dimension used for the isotropic symmetry. Only one dimension is presented, but the other two dimensions were generated using the same distribution.

- The **dimensions** were randomly generated by using a uniform distribution with 10 μm as the lower limit and 1 cm as the upper limit, considering that these are

typical dimensions of the crystals used for RUS measurements in the *Quantum Materials Lab*.

The 150,000 data sets for each symmetry was generated using 50,000 data sets for each of the three different types of sample shapes considered: rectangular parallelepiped, ellipsoidal cylinder and spheroid. With this, the forward problem was solved using order 10 polynomials (Cartesian power series) and 40 resonance frequencies were obtained per iteration. Thus, each entry in the data-set had the density, the shape, the dimensions, the independent elastic constants, and the resonance frequencies for each sample being simulated.

3.2 Pre-processing the data-set

Pre-processing the data-set is one of the most important stages in machine learning and in data science in general [6]. Considering this, to pre-process the data-set generated in 3.1, the following steps were conducted.

First, it is important to mention that each subset of the data-set was pre-processed individually, but the procedure was exactly the same for each. Considering the experimental uncertainties of the RUS measurements, the fact that each resonance frequency has a width associated to it (their shape is Lorentzian), and possible errors in the determination of the central frequency of each resonance, a gaussian noise was added to the frequencies determined in the forward problem. The gaussian noise for each resonance was centered at each frequency, and had a standard deviation of 1 kHz, as this is the typical value for the experimental bandwidth in such measurements. This step was very important since the data-set should be thought of as synthetic data, so including a gaussian noise can give the data-set a more realistic approach. As it was presented in section 2.2.2, not always each resonance is excited for a given sample, meaning that in the experimental portion of RUS, having missing frequencies can be a common situation. To deal with this, a re-sampling procedure was done to each subset on the data-set. This consisted of a re-sampling of each iteration going from one iteration with 40 resonance frequencies to 10 sub-iterations with 10 frequencies each. This allowed the machine learning solutions to be robust towards missing frequencies (refer to section 4.6). In figure 15 a description of this re-sampling process can be seen.

Having done this, a simple fit with a stock random forest was done, giving the model the density, shape, dimensions, and frequencies as inputs, and expecting the respective elastic constants as outputs. This was done in order to calculate the importance of each feature in the input, and to understand if some of them might be contributing less than others. With this, it was found that the shape of the sample

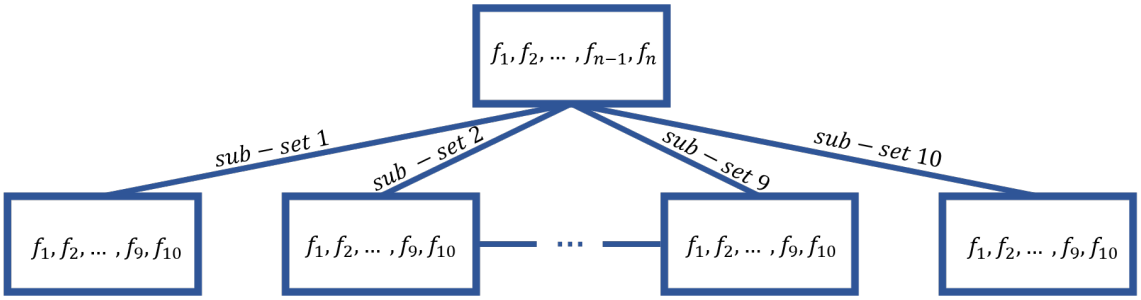

Figure 15: Re-sampling procedure for each of the 150,000 data sets. Each iteration of 40 frequencies was sub-sampled to 10 iterations of 10 random frequencies each.

and most of the frequencies were not highly correlated to the outputs. So, the shape was disregarded, and a principal component analysis was done on the frequencies in order to project the frequencies to the component that explains most of the variance of the spectrum. With this, the same stock random forest was used to calculate once again the importance of the new inputs which in this case were the density, dimensions and the first principal component of the frequencies. The importance of each of these features was found to be significant, so these features were considered as the input of each model for the rest of the project.

3.3 Training, testing, and optimizing deep learning solutions

In order to find the best model for the problem at hand, different deep learning solutions were compared. For this first step, regarding the search for the best model to predict the elastic constants, only the isotropic symmetry was used. This was done to consume less time and considering that the only difference between symmetries is the number of independent elastic constants, repeating this procedure for each symmetry would be inefficient. Thus, to train the deep learning solutions, each subset (after being pre-processed) was divided into training and testing sets (70% of the data as training and 30% of the data as test) for the regression trees; and training, testing and validation for the neural networks (40% of the data as training, 30% as validation and 30% of the data as test). This is due to the fact that the regression trees are more robust to over-fitting the data, while for the neural networks it is important to monitor the validation sets to prevent over-fitting [6].

Having divided each subset, the regression trees were implemented first. For the regression trees, a random forest regressor was used as the model to be trained. Few variations on the hyperparameters were done to understand if this model would be appropriate. Having tested some configurations and understanding the capabilities of this model, two different neural networks were implemented to compare the different

deep learning solutions. The first architecture implemented was a simple sequential neural network, where the different number of layers and neurons, the number of epochs, the loss function and the optimizer were experimented with to find the ideal hyperparameters of the model. After having found a satisfactory architecture, the complexity of the neural network was increased by adding convolutional layers in order to accelerate the convergence of the loss in the algorithm. Different iterations of the convolutional layers were tested until arriving to a satisfactory model.

The results obtained with the neural networks were significantly less accurate than those obtained with the regression trees. Considering this, the random forest regressor was selected as the final model for each symmetry. Having selected the appropriate model for the inverse problem, a thorough study of the hyperparameters was done. To do this, a randomized grid search including a 3 *k-fold* cross validation was done. This hyperparameter search routine was chosen since evaluating random configurations can speed up the computation time when evaluating the hyperparameter grid, and the *k-fold* cross validation can prevent over-fitting in the random forest regressor.

3.4 Comparing with traditional solutions

After implementing and selecting the final model, the results obtained with the final model were compared with the traditional solutions. The results were only compared to the traditional solution described in section 2.2.1 as this is the solution that has been regularly used in the *Quantum Materials Research Group*, meaning that it was readily available. Thus, the results obtained with the deep learning solution were compared to the Levenberg-Marquadt Algorithm [24] in both terms of accuracy and computation time.

To compare the accuracy between models, the next procedure was followed. Since the deep learning models predict the elastic constants from the resonance frequencies, the predicted elastic constants were fed through the forward problem in order to obtain the predicted resonance frequencies, and the predicted resonance frequencies were compared with the real resonance frequencies that were used to train the models (clearly, selecting only data from the test set). This is important since the Levenberg-Marquadt Algorithm [24] presents the error associated to each iteration in terms of the difference between the measured frequencies and the frequencies obtained through the estimation of the elastic constants. This allowed the comparison to be appropriate since the same error was being analyzed in both solutions. So, in order to analyze the accuracy in terms of error, the traditional solution was used to test the same elastic constants that were tested in the deep learning model.

Finally, in order to compare the computation time, a similar procedure was used. First, the computation time for the deep learning solutions was calculated (the train-

ing time was not taken into account since training is only done once) as the time it takes for the model to, given the input data, return the output data i.e., the predicted elastic constants. Having determined the time taken for the deep learning solutions to present the predicted output, the computation time for the traditional solution was calculated. This was done by taking the time between the start of the algorithm until the last iteration where the error is satisfactory.

3.5 Testing on experimental data

Having implemented the final models, these were tested on real experimental data including the three symmetries used throughout the project. Comparing with the results obtained by the traditional solution when analyzing the real experimental, it was found that the final models were not as accurate on the experimental data, so the models were trained once again on a more appropriate data-set. Having done this, the new models were tested and compared to the results obtained with the final models from the last sections and with the traditional solution.

3.6 Design and implementation of a graphic user interface

To make use of the models implemented, a graphic user interface (GUI) was both designed and implemented. To do this, different design iterations were studied until arriving to the final GUI which was implemented in Python using the library Tkinter. It is important to mention that the GUI takes the resonance spectrum, dimensions and density of the sample, and shows both the predicted elastic constants and the error between the predicted resonance frequencies and the experimental frequencies as well as outputting a file with the predicted resonance frequencies.

4 Results and Analysis

4.1 Forward Problem Implementation

As it was mentioned in the methodology, the implementation of the forward problem had to be tested to ensure that it was done appropriately. To do this, the program written was tested on real measurements of $La_{2-x}Sr_xCuO_4$ and $SrTiO_3$ that were performed at the National High Magnetic Field Laboratory in Tallahassee, Florida [22]. The elastic constants, dimensions and densities of each material were given as inputs to the forward problem implementation, and the resulting resonance frequencies were compared to those found experimentally. In total, five different measurements were used for the comparison, averaging an error of (1.198 ± 1.436) % between the frequencies calculated and the frequencies measured experimentally. This result showed that the implementation of the forward problem was done correctly and with this, the dataset was generated as described in section 3.1.

4.2 Preliminary Analysis

As a preliminary analysis, having pre-processed the dataset, a stock random forest regressor was fitted in order to understand the importance of each feature in the input with respect to the output. So, for the first iteration, the regressor was fitted by inputting the density, shape, dimensions, and frequencies, expecting the elastic constants as outputs. The results for the importance of each feature can be seen in figure 16.

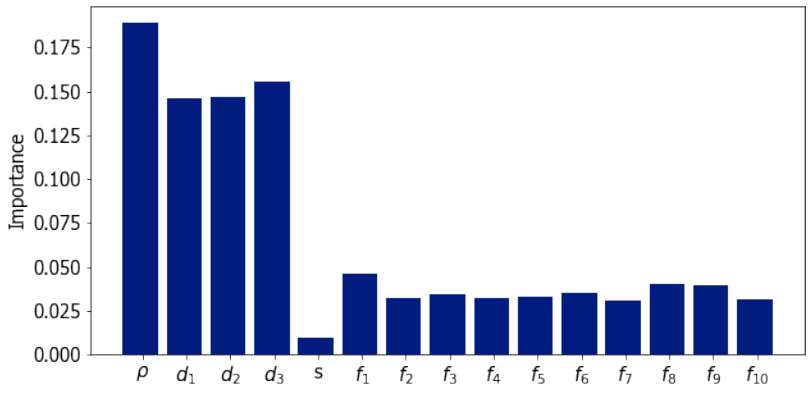

Figure 16: Importances of the first input vector calculated through the stock random forest regressor. s represents the shape of the material.

From figure 16 it is clear that both the shape and most of the frequencies have a low importance when predicting the output. Considering this, the shape was disregarded completely, but the frequencies were pre-processed once again since they carry essential information when predicting the elastic constants (as seen in the forward problem derivation). A principal component analysis was done on the frequencies and as the first component explained most of the variance for the frequencies (0.748 percent of the variance was explained by the first principal component), the input was changed to be only the density, dimensions, and the frequencies projected onto the first principal component of the PCA analysis. With this, the same random forest regressor was fitted, and the importance of each feature in the input was calculated again. The values obtained can be seen in figure 17.

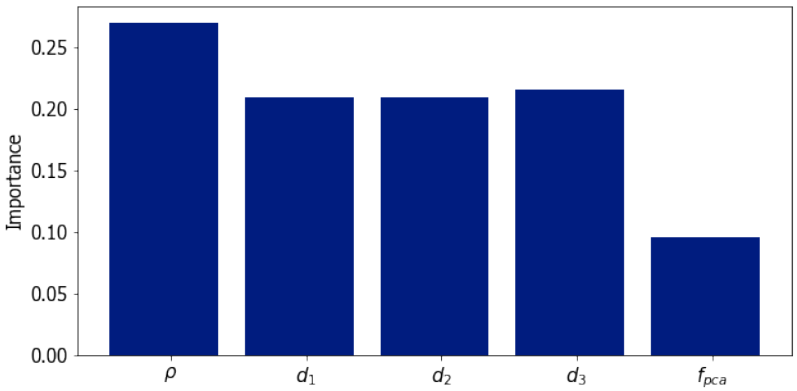

Figure 17: Importances of the second input vector calculated through the stock random forest regressor.

As each of the features presented in figure 17 actually have a significant importance when predicting the elastic constants, this was the input that was used for all the models in the following analysis. Having determined the best input for the problem at hand, it was necessary to understand which deep learning strategy might be more appropriate as an alternative to the traditional solutions used to solve the inverse problem in resonant ultrasound spectroscopy. As it was mentioned in the methodology, three different models were tested including a random forest regressor, a sequential neural network, and a convolutional neural network. The variation of the hyperparameters of each model for this stage was done manually to grasp the potential of each solution in predicting the elastic moduli. This process was done only for the isotropic symmetry, under the hypothesis that the same model that worked for the isotropic symmetry would work for the others. This was a valid approximation since the only thing that changes from symmetry to symmetry is the number of independent elastic moduli.

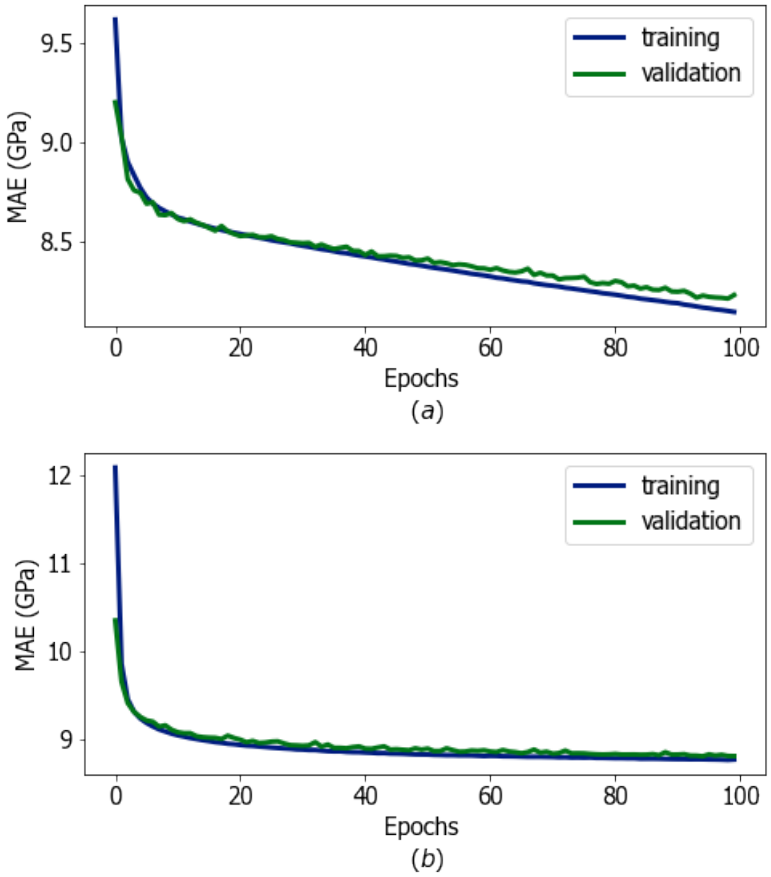

Figure 18: Mean absolute error as a function of the epochs for the neural networks. (a) shows the result for the sequential neural network and (b) for the convolutional neural network.

Table 1: Comparison of the results obtained with the different deep learning strategies used. MAE refers to mean absolute error and MAEP to mean absolute percentage error between the predicted and real elastic constants.

Model	**MAE (GPa)**	**MAEP (%)**
Random Forest Regressor	(0.726 ± 3.404)	(8.732 ± 50.374)
Sequential Neural Network	(8.228 ± 17.990)	(70.908 ± 156.323)
Convolutional Neural Network	(8.698 ± 18.402)	(73.499 ± 146.786)

In table 1 the results obtained for each model described above can be found. The results are reported in terms of the mean absolute error and mean absolute percentage error between the real elastic moduli and the ones predicted by each model, clearly

all evaluated on the test set. As it can be seen from table 1, the random forest regressor outperformed both the sequential neural network and the convolutional neural network by approximately 8 GPa when looking at the mean absolute error and about 60% referring to the mean absolute percentage error. It is possible that this difference is because random forest regressors are a more appropriate solution to low dimension regression problems since they are **off-the-shelf** models [15]. Regarding the neural networks, it is important to mention that while the result obtained with the sequential neural network and the convolutional neural network is remarkably similar, the error tends to converge faster for the convolutional neural network since it contains convolutional layers which are proven to speed up convergence [6]. In figure 18 the mean absolute error as a function of the epochs for both the sequential neural network and the convolutional neural network is shown. From this figure it can be seen that the mean absolute error tends to converge faster for the convolutional neural network, meaning that the error reaches its minimum in less epochs. But even though the convergence of the convolutional neural network is faster than that of the sequential neural network, the random forest regressor was found to be the best solution as an alternative to the traditional models used to solve the inverse problem.

As a reminder, it is important to mention that the errors obtained in table 1 were only used to select the appropriate model for solving the inverse problem. Meaning that this error is only for comparison purposes. Refer to the next sections to see the minimized errors for each symmetry studied.

4.3 Isotropic Symmetry

To find the best hyperparameters for the model selected, in this case a random forest regressor, a more appropriate hyperparameter search was done. To do this, a randomized grid search was used where the grid parameters implemented for the search routine were:

```
param_distributions={'bootstrap': [True, False],
                     'max_depth': [None],
                     'max_features': ['auto'],
                     'min_samples_leaf': [1, 2, 4],
                     'min_samples_split': [1, 2, 4],
                     'n_estimators': [int(x) for x in np.linspace(5,200,20)]},
```

By using a randomized search including a 3 *k-fold* cross validation the best hyperparameters found for the isotropic symmetry were:

```
{'n_estimators': 179, 'min_samples_split': 2, 'min_samples_leaf': 1,
'max_features': 'auto', 'max_depth': None, 'bootstrap': False}
```

It is important to mention that the scoring metric used for this hyperparameter search was the mean absolute error between the predicted and the real outputs. This specific configuration obtained a mean absolute error of (0.425 ± 4.873) GPa when evaluating the error on the test set, which translated to a mean absolute percentage error of around 4%. In figure 19, the results for the predicted constants and the true constants can be seen. It is important to mention once again that the results presented on these figures are from the test set, meaning that the model had never seen the data before it was tested.

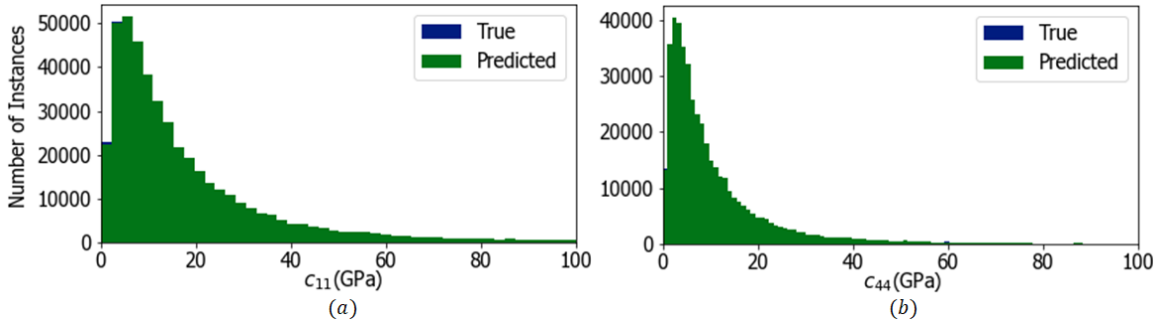

Figure 19: Difference between the predicted values and the true values for the isotropic symmetry. (a) shows the difference for c_{11} and (b) for c_{44}.

From these figures, it can be said that the model is successful in determining the elastic constants for this specific symmetry since the difference between the predicted values and the true values is low (quantified as (0.425 ± 4.873) GPa when evaluating the mean absolute error). Even though the model seemed as it can predict the values for the elastic moduli with a high accuracy, it was important to see how this solution compared to the traditional solution that has been used and was described in section 2.2.1. In order to compare the errors, the following metric was used:

$$\text{RMSE } \% = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{f_{real} - f_{predicted}}{f_{real}}\right)^2} \times 100 \tag{52}$$

The real frequencies obtained through the forward problem represented the experimental frequencies in the metric presented above. This allowed to use the same data when evaluating the deep learning model and the traditional solution. Thus, with the predicted outputs from the deep learning solution, a predicted set of frequencies was calculated and compared to the set of frequencies obtained with the real

outputs. As for the traditional solution, since it is dependent on the initial guesses for the elastic moduli, this metric was calculated for different elastic moduli guesses to understand its robustness to poor initial guesses. The initial guesses for each of the elastic moduli were varied ranging from 0.5 of their real values up to their exact values. And with these initial guesses, the predicted frequencies were compared to the real frequencies as mentioned before. Table 2 summarizes the results obtained for the traditional solution. To calculate these results, 30 samples from the test set were randomly selected and evaluated.

Table 2: Results obtained for the traditional solution evaluated on the isotropic symmetry. For $0.5c_{true}$ the traditional solution fails to converge.

Initial Guess	**Root Mean Squared Error (%)**
c_{true}	(15.907 ± 20.479)
$0.9c_{\text{true}}$	(17.332 ± 20.845)
$0.8c_{\text{true}}$	(21.265 ± 25.106)
$0.5c_{\text{true}}$	💣

From these results, there are several things that should be mentioned. First, as the initial guesses stir away from the real values, the error associated to the traditional solution increases. As well as this, the variance of the results from the traditional solution is high, even averaging more than the error itself. This is clearly problematic when a precise solution is needed. It is also important to indicate that out of the 30 samples that were used to test the traditional solution, for the initial guesses of c_{true} and $0.9c_{\text{true}}$ only 17 samples converged to a solution, and for the initial guess of $0.8c_{\text{true}}$ 12 out of the 30 samples converged. This once again shows that the traditional solution is highly dependent on the initial guesses for the elastic moduli.

Now, looking at the results obtained with the deep learning strategy implemented, a root mean squared percentage error of (0.332 ± 1.786) % was obtained. Clearly, the random forest regressor outperforms the results obtained with the traditional solution in terms of both average error and variance. The error obtained with the random forest regressor is about one order of magnitude lower and the variance as well. This shows the potential that the use of tree-based regressions can have with the problem at hand.

Finally, it is imperative to comment on the computation time needed to run both solutions. The deep learning solution took (0.206 ± 0.071) seconds to, given the input data, calculate the elastic moduli. This, without considering the training time needed to fit the training data to the random forest regressor. While for the traditional solution, using order 10 polynomials, each iteration took (0.14 ± 0.03) seconds. Throughout the testing of this solution, it was observed that the model

tended to converge after 100 iterations (not considering the times that the model failed to converge), meaning that in average, the traditional solution presented its results in around (14 ± 3) seconds. With these results, the deep learning model has a clear advantage of about two orders of magnitude when comparing the computation time needed to run both solutions.

4.4 Cubic Symmetry

As well as for the isotropic symmetry, to find the best hyperparameters for the selected model, a randomized search was done. The grid parameters used for the search routine were the same as the ones used for the isotropic symmetry. Thus, by using a randomized search including a 3 *k-fold* cross validation the best hyperparameters found for the cubic symmetry were:

```
{'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1,
'max_features': 'auto', 'max_depth': None, 'bootstrap': False}
```

This specific configuration obtained a mean absolute error of (1.161 ± 6.847) GPa when evaluating the mean absolute error on the test set. In figure 20 the results for the predicted constants and the true constants can be seen.

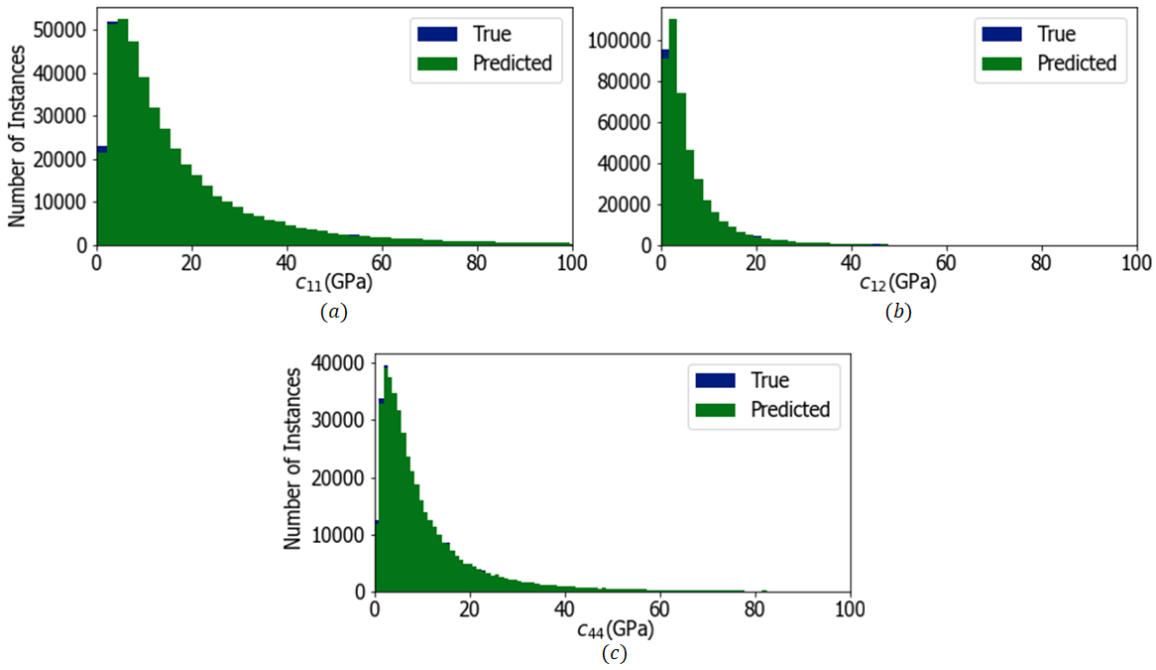

Figure 20: Difference between the predicted values and the true values for the cubic symmetry. (a) shows the difference for c_{11}, (b) for c_{12} and (c) for c_{44}.

As well as for the isotropic symmetry, it can be said that the model is successful in determining the elastic constants for this specific symmetry since the difference between the predicted values and the true values is low when studying the figures presented above (quantified as (1.161±6.847) GPa when evaluating the mean absolute error). For this symmetry, the same procedure done in the isotropic symmetry to compare the deep learning results to the traditional solutions was performed. Table 2 summarizes the results obtained for the traditional solution.

Table 3: Results obtained for the traditional solution evaluated on the cubic symmetry. For $0.5c_{true}$ the traditional solution fails to converge.

Initial Guess	**Root Mean Squared Error (%)**
c_{true}	(8.831 ± 8.516)
$0.9c_{\text{true}}$	(10.588 ± 8.796)
$0.8c_{\text{true}}$	(19.768 ± 19.335)
$0.5c_{\text{true}}$	💣

Once again, as the initial guesses stir away from the real values, the error associated to the traditional solution increases. As well as this, the same behavior is found for the variance of the results since the variance averages to as high as the error itself. Out of the 30 samples that were used to test the traditional solution in this specific symmetry, for the initial guess of c_{true} 22 samples converged to a solution, for $0.9c_{\text{true}}$ 19 samples converged, and for the initial guess of $0.8c_{\text{true}}$ the same 19 samples converged to a solution. Even though the convergence rate for this symmetry is different than the results obtained for the isotropic symmetry, with the results shown in table 3 it is still clear that the traditional solution is highly dependent on the initial guesses for the elastic constants.

Looking at the results obtained with the deep learning solution implemented for this symmetry, with the random forest regressor used, a root mean squared percentage error of (0.418 ± 1.611) % was obtained. Clearly, the deep learning solution outperforms once again the traditional solution in terms of both average error and variance. The error obtained with the random forest regressor is about one order of magnitude lower and the variance as well. This validates the initial hypothesis mentioned in the preliminary analysis, where the same model used for the isotropic symmetry can be used for different symmetries (but with different hyperparameters).

Finally, it is important to comment on the computation time needed to run both solutions for the cubic symmetry. The deep learning solution took (0.164 ± 0.061) seconds without considering the training time as mentioned before. While for the traditional solution, using the same order 10 polynomials as for the isotropic case, each iteration took around 0.14 seconds. Averaging convergence after 100 iterations,

the traditional solution presented its results in around 14 seconds. Comparing these results, the deep learning model has an advantage of about two orders of magnitude in terms of computation time.

4.5 Tetragonal Symmetry

As well as for the other symmetries, to find the best hyperparameters for the model selected, the same randomized search was done. The grid parameters used for the search routine were the same as the ones used for the last two symmetries. Therefore, by using a randomized search including a 3 *k-fold* cross validation the best hyperparameters found for the tetragonal symmetry were:

```
{'n_estimators': 189, 'min_samples_split': 2, 'min_samples_leaf': 1,
'max_features': 'auto', 'max_depth': None, 'bootstrap': False}
```

This specific configuration obtained a mean absolute error of (0.461 ± 4.469) GPa when evaluating the mean absolute error on the test set. In figures 21 and 22, the results for the predicted constants and the true constants can be seen.

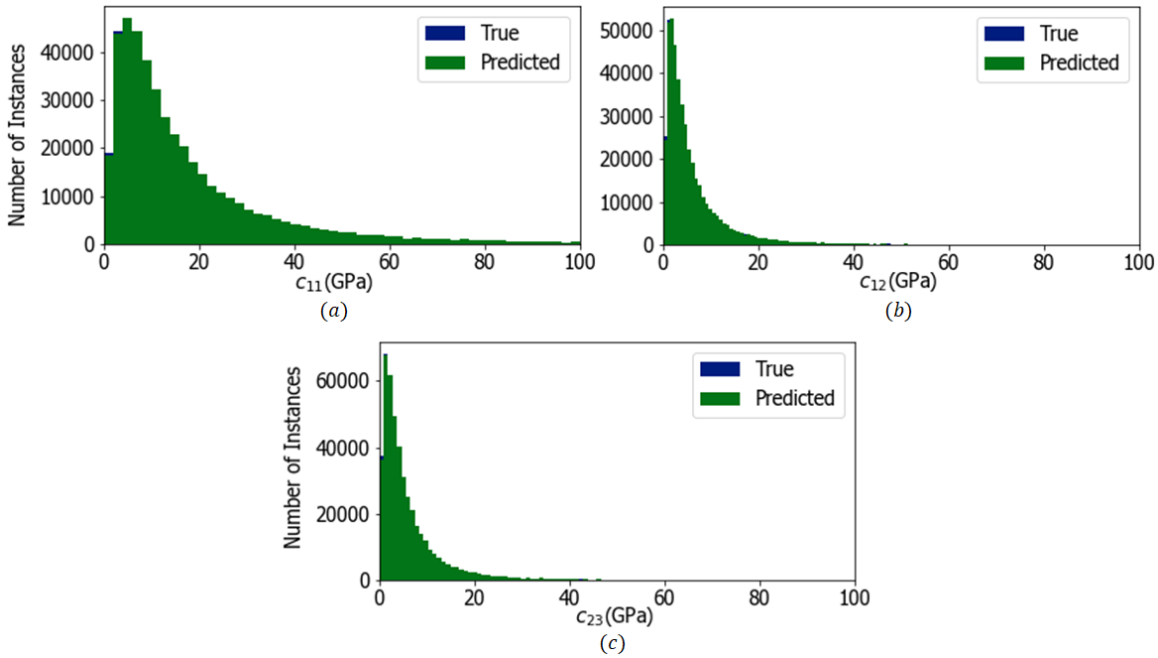

Figure 21: Difference between the predicted values and the true values for the tetragonal symmetry. (a) shows the difference for c_{11}, (b) for c_{12} and (c) for c_{23}.

Once again, by looking at the figures presented above, it can be said that the model is successful in determining the elastic constants for the tetragonal symmetry since the difference between the predicted values and the true values is low (quantified as (0.461 ± 4.469) GPa when evaluating the mean absolute error). For this symmetry, the same procedure to compare the deep learning results to the traditional solutions was performed. Table 2 summarizes the results obtained for the traditional solution evaluated in the tetragonal symmetry.

Table 4: Results obtained for the traditional solution evaluated on the tetragonal symmetry. For $0.5c_{true}$ the traditional solution fails to converge.

Initial Guess	**Root Mean Squared Error (%)**
c_{true}	(23.640 ± 18.509)
$0.9c_{\text{true}}$	(25.599 ± 16.513)
$0.8c_{\text{true}}$	(26.580 ± 20.537)
$0.5c_{\text{true}}$	💣

The error obtained for this specific symmetry is higher than the ones obtained for the isotropic and cubic symmetries. This might be due to the difference in independent elastic moduli, for which in the case of tetragonal symmetry, there are 6 independent variables that need to be predicted. But just as for the other symmetries, the same behavior is seen, where if the initial guesses stir away from the real values, the error associated to the traditional solution increases. In this specific case, out of the 30 samples that were used to test the traditional solution, for the initial guesses of c_{true} and $0.9c_{\text{true}}$ only 9 samples converged to a solution, and for the initial guess of $0.8c_{\text{true}}$ 7 out of the 30 samples converged. This shows that the traditional solution struggles when trying to predict the elastic moduli for the tetragonal symmetry.

Considering the results obtained with the deep learning, a root mean squared percentage error of (0.579 ± 2.169) % was obtained. Comparing with the traditional solution, the deep learning solution outperforms it in terms of both average error and variance. The error obtained with the random forest regressor is about one order of magnitude lower and the variance as well. It is important to mention that the results obtained with the deep learning solution for the different symmetries oscillates between similar values, which is also an improvement with respect to the traditional solution.

To finish, it is important to comment on the computation time needed to run both solutions. The deep learning solution took (0.139 ± 0.031) seconds without considering the training time. While for the traditional solution, each iteration took around 0.14 seconds per each iteration. Using 100 iterations to reach convergence, the traditional solution presented its results in around 14 seconds once again. This is

an interesting added value of the traditional solution since regardless of the symmetry being studied, the time it takes to converge is similar. The same can be said for the deep learning solution, since the computation time needed to reach the solution is almost invariant to the symmetry being explored.

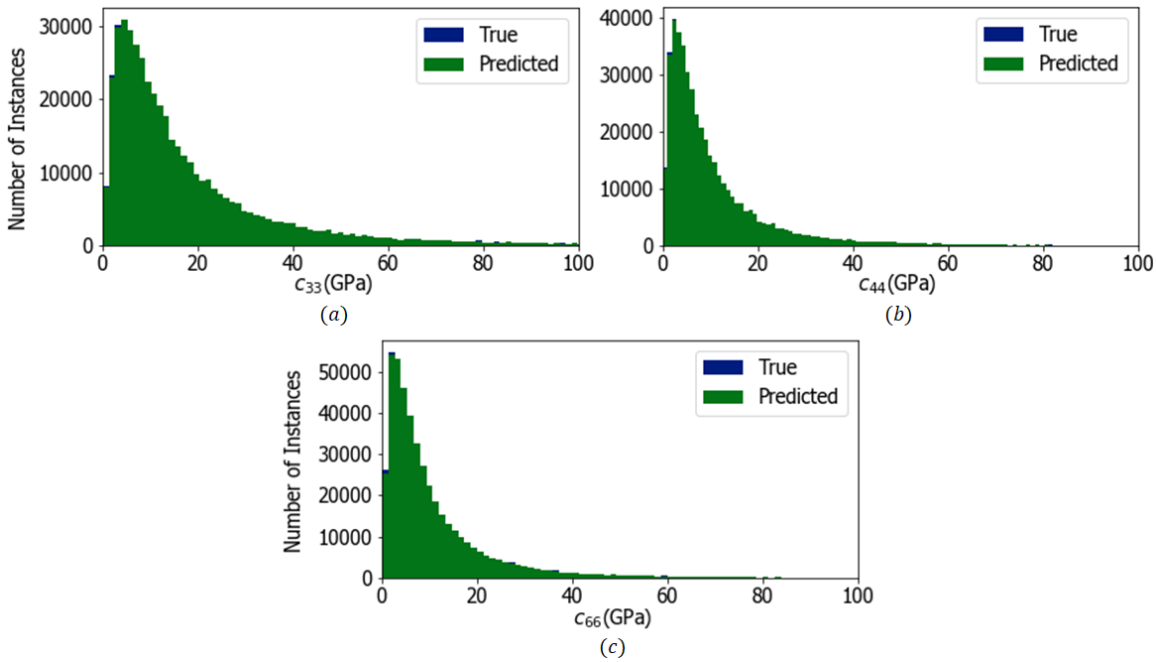

Figure 22: Difference between the predicted values and the true values for the tetragonal symmetry. (a) shows the difference for c_{33}, (b) for c_{44} and (c) for c_{66}.

4.6 Test on Experimental Data

With the trained models, each was tested on real experimental data. Multiple resonance spectrums for different materials were provided by Dr. Albert Migliori (physicist who first introduced the term Resonant Ultrasound Spectroscopy [25], from Los Alamos National Laboratory, USA). As expected, the models presented in sections 4.3, 4.4 and 4.5 were found to be highly dependent on the data used to train them. This can be seen in tables 5, 6 and 7 where the root mean percentage error of the elastic constants (of the real materials) obtained by using the initial model is significantly higher than both the previous results shown and the results obtained with the traditional solution. Thus, even though the trained models outperformed the traditional solution in the previous sections, it is clear that the training data used to train these models is skewed towards low values of the elastic moduli.

With this in mind, the models were trained once again, but using data centered

towards the real elastic moduli of the materials used in this section, which will be referred to as "centered dataset" from now on. To do this, the mean and the standard deviation of the samples for each symmetry were calculated and the elastic constants used to generate the training dataset were taken from a positive normal distribution with mean (μ) and standard deviation (σ) corresponding to the experimental values found for each symmetry. The values for each distribution were as follows (all numbers are given in GPa):

- **Isotropic Symmetry:** $c_{11} \rightarrow (\mu = 209, \sigma = 36)$ $c_{44} \rightarrow (\mu = 57, \sigma = 14)$
- **Cubic Symmetry:** $c_{11} \rightarrow (\mu = 256, \sigma = 137)$ $c_{12} \rightarrow (\mu = 75, \sigma = 50)$ $c_{44} \rightarrow (\mu = 67, \sigma = 30)$
- **Tetragonal Symmetry:** $c_{11} \rightarrow (\mu = 230, \sigma = 24)$ $c_{12} \rightarrow (\mu = 70, \sigma = 25)$ $c_{23} \rightarrow (\mu = 83, \sigma = 24)$ $c_{33} \rightarrow (\mu = 223, \sigma = 34)$ $c_{44} \rightarrow (\mu = 67, \sigma = 8)$ $c_{66} \rightarrow (\mu = 75, \sigma = 24)$

It is important to mention that the number of samples in the training set for the centered dataset was significantly lower when compared to the original datasets. Remembering that the original datasets had 150,000 samples which where subsampled to 1,500,000 samples, these centered datasets were used with 30,000 samples and subsampled to 300,000 samples. The results obtained with the centered dataset can be seen as well in tables 5, 6 and 7.

Table 5: Root mean percentage error between the predicted and the experimental resonances obtained with the initial model, the centered model and the traditional solution for the isotropic symmetry.

Material	**Initial Model**	**Centered Model**	**Traditional Solution**
$FeNi_{36}$	79.203%	4.406%	1.071%
LPCMO	38.633%	0.874%	0.225%
Nb	46.356%	16.388%	1.007%
Ni	32.932%	17.109%	0.335%
Average	$(49.281 \pm 20.693)\%$	$(9.694 \pm 8.277)\%$	$(0.660 \pm 0.441)\%$

As it can be seen in the three different symmetries, the centered model outperforms the initial model. This proves that the training data is very important when trying to predict the elastic moduli of a sample. It is clear that the model needs to be trained with elastic constants that are near to the experimental values. This can prove to be a limitation of these models since most of the time, the real elastic constants

are unknown. As well as this, there is a high variance between the results obtained with the centered model (of the order of 8%, 6% and 2% for the isotropic, cubic and tetragonal symmetry respectively), which can potentially be fixed by using more training data. Regardless, the centered model is close to the results obtained with the traditional solution for both the isotropic symmetry and the tetragonal symmetry. While for the cubic symmetry, even though the centered model still outperforms that initial model, the error is not comparable to the error obtained with the traditional solution. This might be because of the high σ in c_{11} used to train the cubic model. This also brings to light an important aspect where the deep learning model needs to be trained on a small range of elastic constants.

Table 6: Root mean percentage error between the predicted and the experimental resonances obtained with the initial model, the centered model and the traditional solution for the cubic symmetry.

Material	**Initial Model**	**Centered Model**	**Traditional Solution**
AuZn	43.755%	29.230%	0.908%
EuB_6	77.502%	33.559%	0.186%
FeGa	37.074%	21.508%	0.536%
SmB_6	60.651%	21.260%	0.823%
Average	(54.746 ± 18.127)%	(26.389 ± 6.045)%	(0.613 ± 0.326)%

Table 7: Root mean percentage error between the predicted and the experimental resonances obtained with the initial model, the centered model and the traditional solution for the tetragonal symmetry.

Material	**Initial Model**	**Centered Model**	**Traditional Solution**
$LaSrNdCuO_4$	66.351%	5.972%	11.081%
LSCO	74.124%	2.675%	0.589%
$Sr_3Ru_2O_7$	76.756%	2.343%	0.233%
URu_2Si_2	67.624%	6.041%	0.548%
Average	(71.214 ± 5.024)%	(4.258 ± 2.024)%	(3.113 ± 5.315)%

To test the robustness of the centered models to missing frequencies, one material was taken from each symmetry and it was tested by removing resonances from the experimental data at random. This was done varying the number of missing frequencies until arriving to 5 missing resonances. The results obtained can be seen in table 8 in terms of the predicted elastic constants. As it can be seen, for every single trial (considering the three different materials), the predicted elastic constants were the same

regardless of the number of missing frequencies. The only change that was observed was in the constants for the $Sr_3Ru_2O_7$ in the case of 5 missing frequencies, but the changes are not significant since they are about of 0.001 GPa. This proves that the centered models are indeed robust to missing frequencies in the experimental data. Is is important to mention that, as more frequencies are taken out of the experimental spectrum, the error between the predicted frequencies and the experimental frequencies increases since the experimental frequencies shift in position, but the predicted frequencies are the same for each trial (since the predicted constants do not change).

Table 8: Test of the centered models on different materials including a different number of missing frequencies. M.F. refers to missing frequencies.

$LaPrCaMnO_3$						
Constant	**0 M.F.**	**1 M.F.**	**2 M.F.**	**3 M.F.**	**4 M.F.**	**5 M.F.**
c_{11} (GPa)	303.852	303.852	303.852	303.852	303.852	303.852
c_{44} (GPa)	50.913	50.913	50.913	50.913	50.913	50.913
SmB_6						
Constant	**0 M.F.**	**1 M.F.**	**2 M.F.**	**3 M.F.**	**4 M.F.**	**5 M.F.**
c_{11} (GPa)	332.860	332.860	332.860	332.860	332.860	332.860
c_{12} (GPa)	143.538	143.538	143.538	143.538	143.538	143.538
c_{44} (GPa)	66.291	66.291	66.291	66.291	66.291	66.291
$Sr_3Ru_2O_7$						
Constant	**0 M.F.**	**1 M.F.**	**2 M.F.**	**3 M.F.**	**4 M.F.**	**5 M.F.**
c_{11} (GPa)	227.895	227.895	227.895	227.895	227.895	227.902
c_{12} (GPa)	86.119	86.119	86.119	86.119	86.119	86.497
c_{23} (GPa)	102.999	102.999	102.999	102.999	102.999	102.951
c_{33} (GPa)	209.812	209.812	209.812	209.812	209.812	210.519
c_{44} (GPa)	59.613	59.613	59.613	59.613	59.613	59.624
c_{66} (GPa)	49.500	49.500	49.500	49.500	49.500	49.197

With these results there are three clear advantages when comparing the machine learning strategies to the traditional solution. First, as it was mentioned before, the machine learning solutions do not need initial guesses to predict the elastic constants of the materials being studied, while the traditional solution is highly dependent on the initial guesses as it was shown in tables 2, 3 and 4. Second, this dependence on initial guesses makes the traditional solution hard to use, but most importantly, the results obtained can have a very high variance. This can be seen comparing the results shown in tables 2, 3 and 4 were the error is significantly higher than that

shown in tables 5, 6 and 7. The machine learning solutions lower this variance in the results, given that they are trained with the appropriate data-sets such as the centered models discussed before. And third, the traditional solution is not robust to missing frequencies in the experimental spectrum, while the machine learning models presented are, as shown in table 8. Finally, while the error in tables 5, 6 and 7 favor the traditional solution, it is important to take into account that these were achieved through multiple iterations by varying the initial guesses until arriving to a satisfactory solution. The machine learning models remove the need for iterations and present the result in one simple use, and the error achieved with these solutions suggest that a comparable error can be reached considering the error obtained with the traditional solutions.

4.7 Graphic User Interface

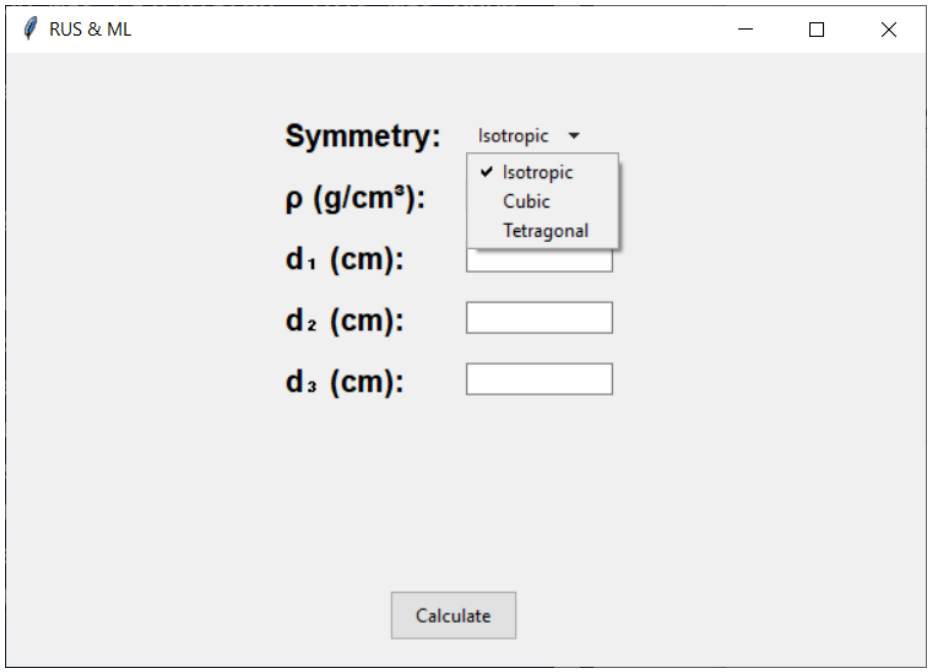

Figure 23: First window when using the graphic user interface.

The actual GUI is very simple and was coded using Tkinter in Python. In figure 23, the first window that appears when running the GUI can be seen. The GUI is composed of 6 different input buttons. The first five are for the user to input the symmetry of the material being studied (this is implemented through a drop-down menu with the available models), the density of the sample that is being studied and the dimensions of such sample. It is important to note that the dimensions should be

entered as length - width - depth (order not important) for a parallelepiped, major diameter - minor diameter - length (order not important) for an ellipsoid cylinder and major diameter - submajor diameter - minor diameter (order not important) for a sphere. In figure 24 the data inputting can be seen.

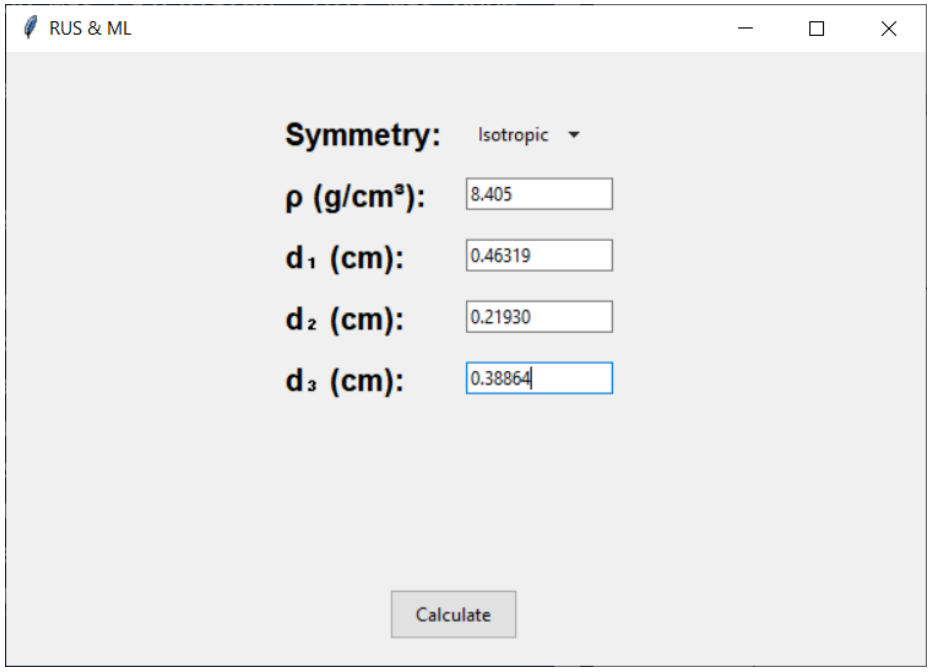

Figure 24: Data inputting for the graphic user interface.

Once the user has inputted all the necessary data, the next step is to press the button "Calculate". Once the user presses this button, a new window appears as seen on figure 25. In this new window, the user must select a .dat or .txt file containing the resonance spectrum in MHz and written as a column. The minimum number of frequencies that needs to be inputted are 10, since the input for the models needs 10 frequencies. If more than 10 frequencies are inputted, the model creates 10 sub-samples of 10 frequencies and outputs the mean elastic constants of each sub-sample (since the models were trained on sub-samples of 10 frequencies, the mean elastic constants have low to no variance as shown in table 8). With this, the GUI back-end uses the models loaded before and presents the elastic constants, as well as the root mean squared percentage error between the experimental frequencies and the predicted frequencies (found by solving the forward problem with the predicted elastic constants). In figure 26 the final result of the GUI can be seen, where the corresponding elastic constants and error are displayed at the bottom. Apart from the elastic constants and the error, the GUI saves a file with the name "freqout.dat" with the predicted resonance frequencies so that the user can compare the experimental spectrum with the one predicted by the models.

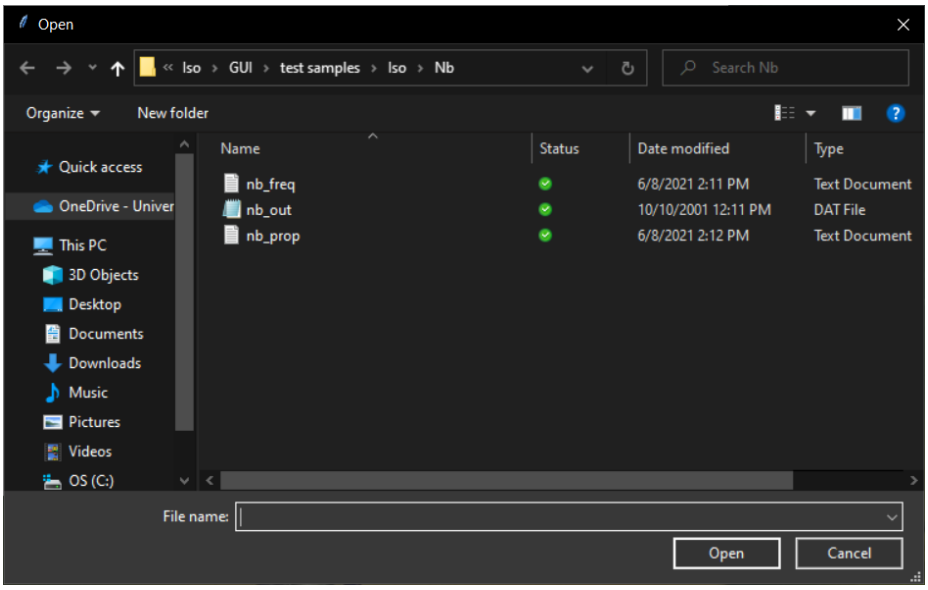

Figure 25: Window opened when pressing the "Calculate" button to select the experimental frequencies.

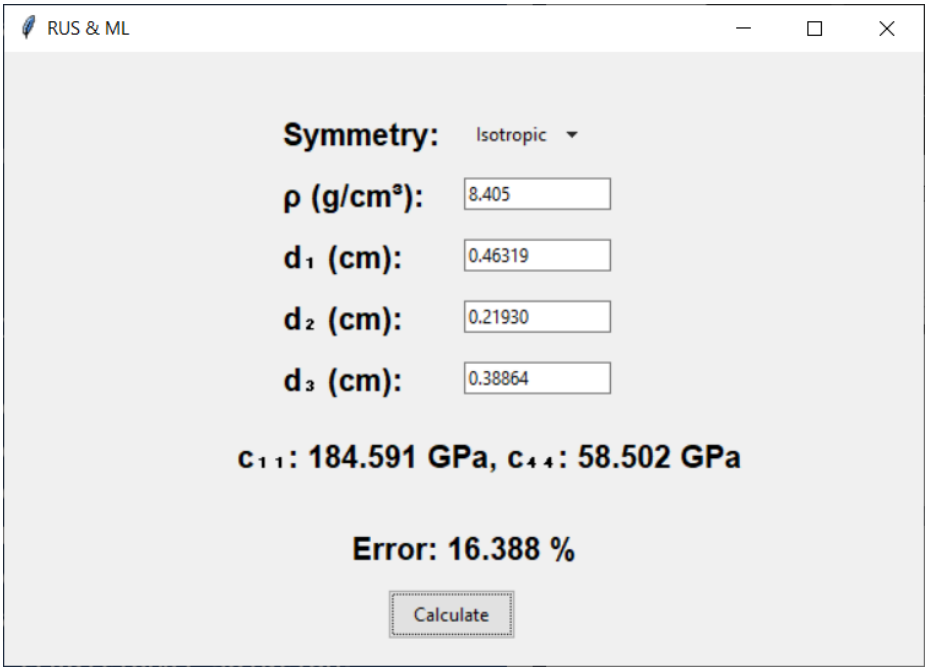

Figure 26: Final result after inputting the properties of the material and the experimental frequencies.

The user can repeat all this process without having to run the script for the GUI

once again, the only thing that has to be done is to erase all the data in the first five spaces, input the new properties of the new material and upload its respective resonance spectrum. And finally, the GUI was coded such that the user can vary the models that the back-end uses to predict the elastic constants. As it was seen in the last section, different models might be needed, so the user can specify which model (or indeed create a new model) to use for each symmetry.

5 Conclusions and Future Work

From the results shown in the last section, there are several things that can be concluded. First, as it was mentioned, a deep learning solution was implemented to be compared with the traditional solution (Levenberg-Marquadt algorithm) used in the inverse problem in resonant ultrasound spectroscopy. To achieve this solution, a dataset was created by solving the forward problem i.e., obtaining the resonance frequencies from the elastic moduli, density, and dimensions of a sample. This dataset was created using synthetic data by gathering the input data from different statistical distributions, and the synthetic data was preprocessed by adding gaussian noise to the outputs obtained (the resonance frequencies).

With the dataset completed, a preliminary analysis was done to define the input data for the deep learning solution, recalling that this solution would solve the inverse problem i.e., predicting the elastic constants from the density, dimensions, and resonance frequencies. To find the optimal input, a stock random forest regressor was implemented and the importance of each feature in the input was calculated. With this process, it was found that the best input consisted of the density of the material, its dimensions and the resonance frequencies projected onto their first principal component (in the context of PCA). Having determined the input for the deep learning strategy, three different models were fitted as a first approach, including a random forest regressor, a sequential neural network and a convolutional neural network. While the convolutional neural network converged much faster than the sequential neural network, the random forest regressor outperformed both neural networks when evaluating the mean absolute error between the real constants and the ones predicted by each model. It is important to mention that this preliminary analysis was only done for the isotropic symmetry and the final model was later generalized for other symmetries.

With the results of the different deep learning strategies, the random forest regressor was chosen as the final model to solve the inverse problem. To find the right hyperparameters, a randomized search including a k-fold cross validation was implemented. This procedure was done for three different symmetries including isotropic, cubic, and tetragonal, which were specifically chosen since the number of elastic moduli is different for each symmetry. Having found the optimal hyperparameters, the model was fitted and compared to the traditional solution. To compare the model to the traditional solution, 30 instances from the test set were randomly chosen and the frequencies calculated with the predicted outputs (the elastic constants calculated with both, the random forest and the traditional algorithms) were compared to the frequencies calculated with the real outputs. Considering this, the deep learning solution outperformed the traditional solution in both terms of error and variance. For the three different symmetries studies, the deep learning solution presented an error

and variance of two orders of magnitude less than the traditional solution. In addition, the traditional solution showed problems with convergence as the initial guesses for the elastic moduli stirred away from the real values. With this, it was confirmed that the traditional solution is highly dependent on the initial guesses given for the elastic moduli. Finally, the computation time needed to use both solutions was calculated, and for the three symmetries, it was found that the deep learning solution outperformed the traditional solution by two orders of magnitude once again.

The final models were tested on real experimental data. As expected, it was found that the models were skewed towards the data used to train them (low values of elastic constants). Thus, the models were trained once again by using a more appropriate set of training data in the range of the experimental values and the models performed better, approaching to the values of the traditional solution in both the isotropic symmetry and the tetragonal symmetry. Therefore, the results obtained showed that, the deep learning solution has a clear potential in the context of solving the inverse problem in resonant ultrasound spectroscopy. Specifically, three different advantages when comparing the machine learning strategies to the traditional solution were found. First, the machine learning solutions do not need initial guesses to predict the elastic constants of the materials being studied, while the traditional solution is highly dependent on the initial guesses. Second, this dependence on initial guesses makes the traditional solution hard to use and the results obtained can have a very high variance while the machine learning solutions lower this variance in the results. And third, the traditional solution is not robust to missing frequencies in the experimental spectrum, while the machine learning models presented are. With this, the machine learning models remove the need for iterations and present the result in one simple use without the risk of achieving no convergence, and the error obtained with these solutions suggest that a comparable error can be reached considering the order of magnitude of the errors achieved with traditional solutions.

Having tested the deep learning solutions, a graphic user interface (GUI) was implemented to make the process of predicting the elastic constants simpler for inexperienced users. In the GUI, the user inputs the density, dimensions and resonance frequencies of the sample being studied and the program returns the predicted elastic constants and the error between the predicted resonance frequencies (calculated through the forward problem) and the experimental frequencies. As well as this, the GUI outputs a file with the predicted resonance frequencies so that the user can compare them to the experimental frequencies found.

To finish, the potential for future work needs to be discussed. First, it would be interesting to understand the effect that more training data has on each model, this would be useful to understand what the ideal number of samples needs to be in order to decrease training time without sacrificing accuracy. As well as this, a possible solution to the problem with the skewed training samples would be to implement different

random forests for the same symmetry but being trained in different orders of magnitude for the elastic constants. With this, all random forests could be used to predict the elastic constants where the final result would be the output that minimizes the error between the predicted resonance frequencies and the experimental frequencies. Having done this, it would be important to generalize or create different models for the rest of the symmetries that are commonly found in different materials.

References

[1] G. Li and J. Gladden, "High temperature resonant ultrasound spectroscopy: A review," *International Journal of Spectroscopy*, vol. 2010, pp. 206–362, 2010.

[2] B. J. Zadler, J. H. L. Rousseau, J. A. Scales, and M. L. Smith, "Resonant ultrasound spectroscopy: theory and application," *Geophys. J. Int.*, vol. 156, pp. 154–169, 2004.

[3] A. Migliori and J. D. Maynard, "Implementation of a modern resonant ultrasound spectroscopy system for the measurement of the elastic moduli of small solid specimens," *Rev. Sci. Instrum.*, vol. 76, p. 121301, 2005.

[4] F. F. Balakirev, S. M. Ennaceura, R. J. Migliori, B. Maiorov, and A. Migliori, "Resonant ultrasound spectroscopy: The essential toolbox," *Review of Scientific Intruments*, vol. 90, p. 121401, 2019.

[5] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, pp. 431–441, 1963.

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

[7] A. Ajanki, "Differences between machine learning and software engineering," 2018.

[8] M. K. Fig, *Resonant ultrasound spectroscopy in complex sample geometry*. PhD thesis, Montana State University, 2005.

[9] C. Kittel, *Introduction to solid state physics*. Wiley, 1956.

[10] B. J. Ramshaw, A. Shekhter, R. D. McDonald, J. B. Betts, J. N. Mitchell, P. H. Tobash, C. H. Mielke, E. D. Bauer, and A. Migliori, "Avoided valence transition in a plutonium superconductor," *Proc. Nac. Ac. Sci.*, vol. 112, p. 3285, 2015.

[11] K. Madsen, H. B. Nielsen, and O. Tingleff, *Methods for non-linear least squares problems*. Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[12] Y.-T. Kwak, J. Hwang, and C.-J. Yoo, "A new damping strategy of levenberg-marquardt algorithm for multilayer perceptrons," *Neural Network World*, vol. 21, pp. 327–340, 2011.

[13] S. Lathuiliere, P. Mesejo, X. Alameda-Pineda, and R. Horaud, "A comprehensive analysis of deep regression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, p. 2065–2081, Sep 2020.

[14] J. García-Gutiérrez, F. Martínez-Álvarez, A. Troncoso, and J. Riquelme, "A comparison of machine learning regression techniques for lidar-derived estimation of forest variables," *Neurocomputing*, vol. 167, p. 24–31, 2015.

[15] T. Dietterich, "Requirements for off-the-shelf learning methods," 2005.

[16] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: with applications in R.* Springer, 2017.

[17] S. M. H. Hashemi and D. Psaltis, "Deep-learning pdes with unlabeled data and hardwiring physics laws," *arXiv:1904.06578*, 2005.

[18] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[19] S. Ghosh, M. Matty, R. Baumbach, E. D. Bauer, K. A. Modic, A. Shekhter, J. A. Mydosh, E.-A. Kim, and B. J. Ramshaw, "One-component order parameter in uru2si2 uncovered by resonant ultrasound spectroscopy and machine learning," *Science Advances*, vol. 6, no. 10, 2020.

[20] P. Freeman, "Resonant ultrasound spectroscopy - python implementation." `https://github.com/PALab/RUS`, 2015.

[21] F. Giraldo-Grueso and P. Giraldo-Gallo, "Quantum materials uniandes rus." `https://github.com/pgiraldogallo/QuantumMaterialsUniandesRUS`, 2020.

[22] P. Giraldo-Gallo, "Unpublished lsco and srtio3 data," 2015.

[23] G. Gao, K. V. Workum, J. D. Schall, and J. A. Harrison, "Elastic constants of diamond from molecular dynamics simulations," *Journal of Physics: Condensed Matter*, vol. 18, no. 32, 2006.

[24] F. F. Balakirev, "Resonant ultrasound spectroscopy software." `https://github.com/ffb-LANL/resonant-ultrasound`, 2020.

[25] J. Maynard, "Resonant ultrasound spectroscopy," *Physics Today*, vol. 49, no. 1, pp. 26–31, 1996.