1 2 9 0

UNIVERSIDADE Ð COIMBRA

Pedro José Monteiro Alves Carriço

COMPUTATIONAL MATERIALS SCREENING USING MACHINE LEARNING METHODS

Dissertação no âmbito do Mestrado em Física, ramo de Simulação e Modelação Computacional orientada pelo Doutor Tiago Frederico Teixeira Cerqueira e pelo Doutor Márcio Rafael Baptista Ferreira, e apresentada ao Departamento de Física da Faculdade de Ciências e Tecnologia.

Setembro de 2022

Computational materials screening using machine learning methods

Pedro José Monteiro Alves Carriço

Department of Physics
University of Coimbra

A thesis submitted for the degree of
Master in Physics

Coimbra, 20th September 2022

"Do or do not. There is no try!" - Yoda

Agradecimentos

O trabalho apresentado nesta tese, bem como todo o percurso que nela culminou, não teria sido possível sem a contribuição de várias pessoas. Aproveito, então, para lhes agradecer.

Ao Doutor Fernando Nogueira, por me ter mostrado o mundo da simulação em Física da Matéria Condensada.

Aos Doutores Tiago Cerqueira e Márcio Ferreira por me terem guiado pelos caminhos por onde esta tese me levou, pelos conselhos, trocas de ideias e por me terem mostrado o mundo do Machine Learning e as suas potencialidades.

A todos os que, ao longo destes anos, contribuíram para as sessões de procrastinação terapêutica. Certamente que me ia esquecer de alguém se os tentasse nomear.

Aos amigos que o Departamento de Física me deu e aos de sempre por me terem aturado e terem tornado estes anos memoráveis.

À minha tia Leonor e ao meu primo Pedro, pela paciência, ajuda, conselhos e *raspanetes* ao longo destes anos. Sem vocês esta jornada não teria sido a mesma.

Aos meus pais e à minha irmã, por tudo. Por terem estado lá nos bons e maus momentos, pelo apoio e por nunca me terem deixado desistir. Um obrigado não chega.

Abstract

The increase in the number of materials in public databases like Materials Project and the advent of faster GPUs and CPUs made Machine Learning a prominent subject in materials science due to the promise of high accuracy at a much lower computational cost than standard methods. Among the recent state-of-the-art methods in use are crystal graph convolution networks (CGNNs). CGNNs are neural networks that operate on graph-structured data. The network consists of a series of connected nodes, each of which represents a vertex in the graph. The edges of the graph are used to connect the nodes and define the relationship between them.

In recent years, the search for super-hard materials has been a hot topic due to their specific characteristics and possible applications. Most of these searches are performed by screening various materials using Density Functional Theory (DFT) calculations, which, despite being very accurate, have a high computational cost.

In this work, a CGCNN was trained to predict the shear and bulk moduli and Vickers' hardness of a dataset containing 10^6 materials of a recently created dataset. For the most promising candidates, DFT calculations were performed to validate the predictions made by the CGCNN. Young's modulus and Poisson's ratio of the these candidates were also calculated using Mazhnik and Oganov's model. A good agreement between predictions and calculations was observed, and an interesting candidate with Vickers's hardness above 40 GPa was identified.

Resumo

O aumento do número de materiais nas bases de dados públicas como o Materials Project e o aparecimento de GPUs e CPUs mais rápidos tornaram o Machine Learning num tema proeminente em ciência dos materiais devido à promessa de uma elevada precisão com custo computacional inferior ao dos métodos convencionais. Entre os métodos de última geração encontram-se as crystal graph convolution networks (CGNNs). As CGCNNs são redes neuronais que utilizam dados estruturados em grafos. A rede é formada por uma série de nós conectados, onde cada um representa um vértice no grafo. As arestas do grafo são usadas para conectar os nós e definir a relação entre eles.

Nos últimos anos, a busca por materiais super-duros tem sido um tema em destaque devido às suas características específicas e aplicações possíveis. Grande parte destas buscas são efetuadas através da análise de vários materiais recorrendo a cálculos de Density Functional Theory (DFT) que, apesar de serem muito precisos, têm um elevado custo computacional.

Nesta tese foi treinada uma CGCNN para prever o módulo de cisalhamento, o módulo volumétrico e a dureza de Vickers de um dataset com 10^6 de materiais provenientes de um dataset criado recentemente. Para os candidatos mais promissores foram efectuados cálculos de DFT para validar as previsões efectuadas pela CGNN. O módulo de Young e o rácio de Poisson destes candidatos também foram calculados utilizando o modelo de Mazhnik e Oganov. Foi encontrada uma boa concordância entres as previsões e os cálculos tendo ainda sido identificado um candidato de interesse com dureza de Vickers acima de 40 GPa.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

With the advent of faster and more powerful GPUs and CPUs, Machine Learning (ML) has become a major subject of interest in a vast number of areas. From image and speech recognition to financial risk modelling, ML has become a regular presence in everyday life [10, 11].

In materials science, ML methods appear as useful tools to accelerate the design of new materials by predicting their properties with accuracy close to Density Functional Theory (DFT) [2]. The advantage of these ML methods over DFT calculations is their lower computational cost. However, the ML methods should not be seen as a replacement for DFT calculations but as a complementary tool to them [12]. An example of this complementarity is the possibility of using ML to screen a larger number of materials than what is possible to do efficiently with DFT and then validate the predictions of the most promising materials using the standard DFT methods (see, for example, the work done in [1]).

An ML method is only as good as the amount of data available for training it [12, 13]. Therefore, along with the improvement on the computational side, the expansion of computational resources and the development of a stable and reliable theory made the size of public materials databases such as the Open Quantum Materials Database (OQMD) [14], AFLOW [15], Materials Project (MP) [9], or JARVIS [16] grow "exponentially" [12].

As said, the main purpose of ML in materials science is to develop models capable of predicting the properties of a given material. These methods can be divided into two categories according with the input used:

- Compositional input: These models represent a material using its chemical formula alone and, usually, do not consider its structure. This fact can be seen as an advantage, since these models can be used in new materials for which the structural information may not be available [12]. However, some properties of materials depend heavily on structure. For example, in the case of the diamond (hard, wide-band-gap insulator) and graphite (soft, semi-metal) [12] their properties are quite opposite because of the atomic arrangement. In comparison with the structural input models, compositional input models benefit from the fact that databases like SuperCon [17] only contain the chemical formulae of the materials. Examples of these so called compositional input models are the ElemNet [18], the CrabNet [19], and the Atom2Vec [20].

- Structural input: These models use atomic structure and atomic composition information of the material as an input. In this category of models there are some that are mainly based in atomic distances like the crystal graph convolutional neural network (CGCNN) [2, 3], and the materials graph neural network (MEGNet) [21] and there are others like the iCGCNN [22], and the atomistic line graph neural network (ALIGNN) [23] that capture many-body interactions using graph convolution networks (GCN). All these models have in common the use of both the atomic positions and the atomic species as input for the graph neural networks (GNN) which can be a problem when predicting new materials, since these informations are not precisely known. The model [24] presents a possible solution to this by replacing the precise bond distances with embeddings of graph distances. Such implementation allows the model to be applied in studies based on both composition and crystal structure prototype without using relaxed structures as input [12].

There are also models like Finder [25] that can use both composition and structural information as input to predict the properties of materials. For the work done in this thesis the model used was the crystal graph convolutional neural network (CGCNN).

ML methods can also be used in the development of interatomic potentials (IAP). ML removes the "guessing game" involved in the construction of classic IAPs such as Lennard-Jones Potential [26] and the Embedded Atom Method [27]. Then, the application of ML generated IAPs is faster and more accurate than the application of a classic IAP. The problem of a ML IAP is that a large dataset is still required for its training [12]. The ML IAP can be purely based on Neural Networks (NN), like Behler-Parrinello neural network (BPNN) [28] and its variants, or use graph neural network merged with physics-based many-body features of classical IAPs like the model presented in [29].

Finally, other application of ML in materials science is the use of generative adversarial networks (GANs) to perform inverse materials design, i.e, finding materials with suitable properties and act as a complement to forward prediction models. This and other techniques and examples can be found in [12].

1.2 Objectives

The main objective of this thesis is the search for super-hard materials, i.e, to predict the shear and bulk moduli and Vickers' hardness with DFT accuracy of materials from a subset of dataset [1]. The predictions will be made using the CGCNN model [2, 3]. Doing so will require to train and find the best model for each one of the elastic properties and to implement new features on the original code to make it more user-friendly and improve the computational costs. As such, the following steps have to be taken:

- Rewriting of the original code in *Pytorch Lightning* framework [30] and implement a gridsearch method to fine-tune the model parameters using *Optuna* [31];
- Assess the importance of the features of the atom feature vector;
- Train models using the data from [4];
- Use the trained models to predict the shear and bulk moduli and Vickers' hardness of materials from [1] with distance to the convex-hull lower than 200 meV/atom;
- Use the predicted values of Vickers' hardness to search for super-hard materials in the dataset where the predictions were made.

1.3 Thesis' outline

This section aims to give a general idea of the structure of the thesis.

In the second chapter an introduction to elastic properties is given. After introducing the strain and stress tensor, Hooke's law is presented and the expressions for the shear and bulk moduli are derived. Also, since Young's modulus and Poisson's ratio are also quantities of interest for this work, their form is also derived. It follows a discussion about the form of the elastic moduli tensor in crystals and how the number of its independent components changes with the symmetry of the crystal. A small discussion about polycrystals allows for the introduction of a new form of the shear and bulk moduli is introduced. The chapter ends with an explanation of Mazhnik and Oganov's Vickers' hardness model [6].

In the third chapter an overview of neural networks is presented, followed by a explanation of the forward and back propagations and the importance of choosing a good method to update the weights of the network. Convolution neural networks and their key features are then briefly described. In the last section of this chapter the importance of finding the best set of hyperparameters is discussed.

In chapter four, the model used to predict the shear and bulk moduli and Vickers' hardness is introduced. The chapter starts with a description of the original CGCNN [2, 3]. Afterwards, a discussion about the feature vector is presented and it is followed by the changes done in the original code. In the last section the method/program used to perform the hyperparameters optimization is also introduced.

Chapter 5 starts with the characterization of the database used to train the models. Then, the database where the predictions will be made is discussed.

Finally, chapter six starts with the information about the training process, followed by the accuracy of the models trained for each elastic property. The chapter ends with the analysis of the predicted shear and bulk moduli and Vickers' hardness for materials contained in a dataset build from the database from [1]. Young's modulus and Vickers' hardness of these materials is also calculated in this chapter.

In Appendix A are discussed the criteria used in this thesis to evaluate the structural and elastic stability of materials.

Chapter 2

On elastic properties

2.1 The strain tensor

Consider a solid material with some shape and volume that will undergo a deformation. Each point of the undeformed material will be index by its position vector $\mathbf{r} = (x_1, x_2, x_3)$. The application of forces to this material will result in a change in its volume and shape. To study the elastic properties of the material, it is necessary to be able to describe the deformation of the material due to the applied forces in a mathematical way. The deformation will produce a displacement of point $\mathbf{r}$ to a new position $\mathbf{r}' = (x'_1, x'_2, x'_3)$ that can be represented by a displacement vector [5]

$$\mathbf{u} = \mathbf{r}' - \mathbf{r} \Rightarrow u_i = x'_i - x_i. \tag{2.1}$$

The deformation will produce a change in the distance between any two points. If the radial vector joining two infinitesimally close points, before deformation, is $\mathrm{d}\mathbf{x}_i$ the radial vector $\mathrm{d}\mathbf{x}'_i$ that joins these points after the deformation can be written as

$$\mathrm{d}\mathbf{x}'_i = \mathrm{d}\mathbf{x}_i + \mathrm{d}\mathbf{u}_i\,. \tag{2.2}$$

The distance $\mathrm{d}l$ between these two points will change due to the deformation. Then, the modulus of the new distance $\mathrm{d}l'$ after the deformation can be written as [5] (using Einstein's summation convention)

$$\begin{aligned}\mathrm{d}l'^2 &= \mathrm{d}l^2 + 2\frac{\partial u_i}{\partial x_k}\,\mathrm{d}x_i\,\mathrm{d}x_k + \frac{\partial u_i}{\partial x_k}\frac{\partial u_i}{\partial x_l}\,\mathrm{d}x_k\,\mathrm{d}x_l \\ &= \mathrm{d}l^2 + 2u_{ik}\mathrm{d}x_i\mathrm{d}x_k,\end{aligned} \tag{2.3}$$

where the strain tensor u_{ij} can be defined as

$$\begin{aligned} u_{ik} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} + \frac{\partial u_l}{\partial x_i}\frac{\partial u_l}{\partial x_k}\right) \\ &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right) \end{aligned} \tag{2.4}$$

for small deformations.

Since this tensor is symmetric, it can be diagonalized at any given point. It is important to note that if the tensor is diagonalized at some point in the solid material, it will not be diagonal at any other point [5]. Then, at the point where the tensor is diagonal, Eq. 2.3 can be written as

$$\begin{aligned} \mathrm{d}l'^{\,2} &= (\delta_{ik} + 2u_{ik})\,\mathrm{d}x_i\,\,\mathrm{d}x_k \\ &= (1+2u_{11})\,\mathrm{d}{x_1}^2 + (1+2u_{22})\,\mathrm{d}{x_2}^2 + (1+2u_{33})\,\mathrm{d}{x_3}^2 \end{aligned} \tag{2.5}$$

From Eq. 2.5 arises that the strain in any volume element may be regarded as composed of independent strains in three mutually perpendicular directions, namely those of the principal axes[1] of the strain tensor. In addition, if the deformation is small, all the components of the strain tensor are small too [5].

2.2 The stress tensor

In addition to describing the solid material when it undergoes deformation, it is also necessary to describe the interactions between different parts of the solid material. To express the forces that these different parts of the solid material exert on each other, one can use the stress σ. Since two parts of the material can be assumed to be in contact through an imaginary surface, the stress is the force acting on that surface per unit of area [32]. Consider the forces acting on a portion of solid material. The components in direction j of the stress acting on a face perpendicular to the axis i can be written as a tensor σ_{ij} [32]. The schematic representation of these forces is shown in Figure 2.1. The components of stress σ_{ij} can be:

- normal stress components, if $i = j$;
- shear stress components, if $i \neq j$.

[1] The principal axes of the tensor u_{ik} are a set of coordinate axes in which only the diagonal components of the tensor are non-zero [5].

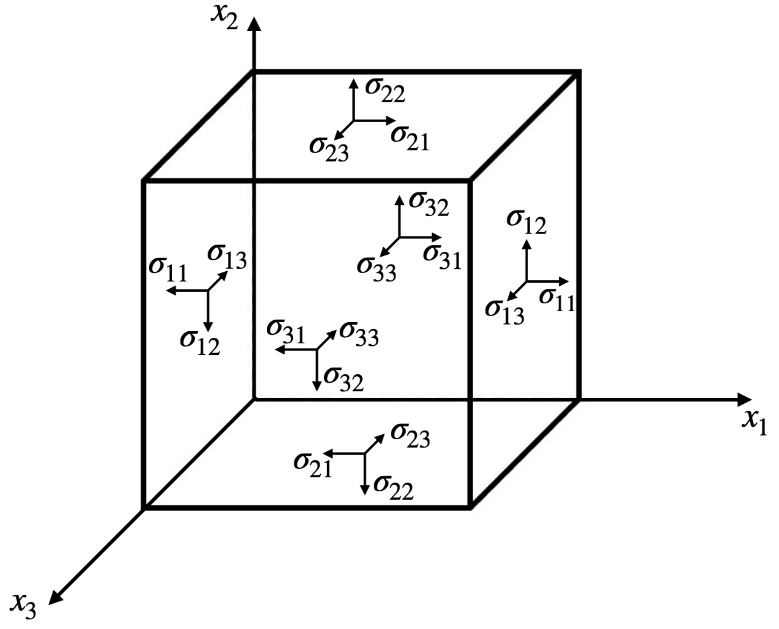


Figure 2.1: Representation of the components of stress on the faces of a cubic solid material.

Once the components of stress are defined, the force **F** acting on an infinitesimal volume of the solid material can be expressed in terms of the components of stress. To do so, consider any two parallel faces of an infinitesimal cube of solid material placed at x_1 and $x_1 + \mathrm{d}x_1$, where $\mathrm{d}x_i$ is infinitesimal distance. The stress component along the x_1 axis on these faces is $\sigma_{11}(x_1)$ and $\sigma_{11}(x_1) + \mathrm{d}x_1$, respectively. If positive stress is defined relative to the outward normal to the face (as in Figure 2.2) these stress components have opposite directions and are shown in Figure 2.2. Then, the component along the x_1 axis of the force acting on these two faces can be written as [32]

$$[\sigma_{11}(x_1 + \mathrm{d}x_1) - \sigma_{11}(x_1)]\,\mathrm{d}x_2\,\mathrm{d}x_3\,. \tag{2.6}$$

Using the same principle, the total force along the x_1 axis acting on the all portion of solid material is

$$\begin{aligned} F_1 = {} & [\sigma_{11}(x_1 + \mathrm{d}x_1) - \sigma_{11}(x_1)]\,\mathrm{d}x_2\,\mathrm{d}x_3 \\ & + [\sigma_{21}(x_2 + \mathrm{d}x_2) - \sigma_{21}(x_2)]\,\mathrm{d}x_1\,\mathrm{d}x_3 \\ & + [\sigma_{31}(x_3 + \mathrm{d}x_3) - \sigma_{31}(x_3)]\,\mathrm{d}x_1\,\mathrm{d}x_2\,, \end{aligned} \tag{2.7}$$

where the three terms represent the force acting on the faces of solid material along the three axes [32]. Eq. 2.7 can be rewritten as

$$F_1 = \left(\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{21}}{\partial x_2} + \frac{\partial \sigma_{31}}{\partial x_3}\right) \mathrm{d}x_1 \, \mathrm{d}x_2 \, \mathrm{d}x_3 \tag{2.8}$$

by making a linear Taylor expansion of $\sigma(x + \mathrm{d}x)$ around x. Defining the x_1-component of the force per unit volume by $F_1^{(v)}$, Eq. 2.8 can be rewritten as

$$F_1^{(v)} = \left(\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{21}}{\partial x_2} + \frac{\partial \sigma_{31}}{\partial x_3}\right). \tag{2.9}$$

Then, by generalisation, the ith component of the force per unit volume due to stress is (using Einstein's summation convention)

$$F_i^{(v)} = \frac{\partial \sigma_{ji}}{\partial x_j} \Rightarrow F_i^{(v)} = \frac{\partial \sigma_{ij}}{\partial x_j}, \tag{2.10}$$

if the stress tensor σ_{ij} is symmetric [5].

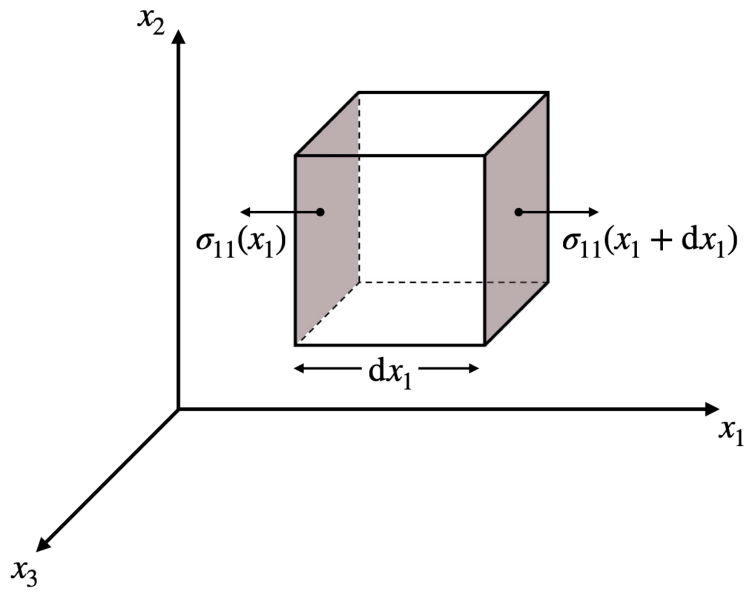


Figure 2.2: The stress on two faces of a cube along the x_1 axis.

2.3 Hooke's law

As mentioned in Sec. 2.1, a body subjected to external forces or stresses undergoes deformation. Therefore, there is a relation between the stress tensor σ and the strain tensor ε. To find this relation consider a deformed solid material and an additional deformation that changes the displacement vector u_i by a small amount δu_i. Assuming an infinitesimal volume $\mathrm{d}V$ of the material, the work δW done by the internal stresses per unit volume is [5]

$$\begin{aligned}\int \delta W \,\mathrm{d}V &= \int \frac{\partial \sigma_{ij}}{\partial x_j}\delta u_i \,\mathrm{d}V = \oint \sigma_{ij}\delta u_i \,\mathrm{d}S_k - \int \sigma_{ij}\frac{\partial \delta u_i}{\partial x_j}\,\mathrm{d}V \\ &= -\frac{1}{2}\int \sigma_{ij}\left[\frac{\partial \delta u_i}{\partial x_j} + \frac{\partial \delta u_j}{\partial x_i}\right]\mathrm{d}V = -\frac{1}{2}\int \sigma_{ij}\delta\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right]\mathrm{d}V \\ &= -\int \sigma_{ij}\delta\varepsilon_{ij}\,\mathrm{d}V\end{aligned} \tag{2.11}$$

From the previous equation it is possible to write the work δW in terms of the change in the strain vector as

$$\delta W = -\sigma_{ij}\delta\varepsilon_{ij}. \tag{2.12}$$

If the deformation is sufficiently small, the solid material returns to the non-deformed state when the external forces causing the deformation are no longer applied to the solid. This type of deformation is called elastic deformation, and all deformations considered until the end of the Chapter will be of this type. If the deformation is large, the solid material will not fully return to its original state when the applied forces are removed. This type of deformation is called plastic deformation [5].

The total work done on a system by external forces is equal to the increase in its energy less the amount of heat absorbed by the system [32]. Ignoring the absorbed heat and considering only the internal energy U, the work done by external forces $\mathrm{d}W = -\delta W$ is equal to the change $\mathrm{d}U$ in the internal energy. Therefore, Eq. 2.12 can be written as

$$\sigma_{ij} = \frac{\partial U}{\partial \varepsilon_{ij}} \tag{2.13}$$

Since all the deformations are considered to be small, it is possible to expand the internal energy U in powers of the strain tensor ε_{ij}. The linear terms of the expansion are zero since [5, 32]

$$\sigma_{ij} = 0 \Rightarrow \varepsilon_{ij} = 0. \tag{2.14}$$

Then, assuming an elastically isotropic solid material[2], the expansion of the internal energy U can be written as a linear combination of the scalars of the second degree of a symmetrical tensor:

$$U = U_0 + \frac{1}{2}\lambda\varepsilon_{ii}^2 + \mu\varepsilon_{ij}^2. \tag{2.15}$$

The parameters μ and λ are the Lamé coefficients.

It is possible to change the shape of the solid material without changing its volume. Such a deformation is called pure shear. The opposite case, i.e change the volume without changing the shape, is called hydrostatic compression. Any deformation can be represented as the sum of a pure shear term $\varepsilon_{ij}^{(ps)}$ and a hydrostatic compression term $\varepsilon_{ij}^{(hc)}$ [5]. The strain tensor becomes

$$\varepsilon_{ij} = \varepsilon_{ij}^{(ps)} + \varepsilon_{ij}^{(hc)} = \left(\varepsilon_{ij} - \frac{1}{3}\delta_{ij}\varepsilon_{kk}\right) + \frac{1}{3}\delta_{ij}\varepsilon_{kk} \tag{2.16}$$

Making use of Eq. 2.16 and neglecting the constant term U_0 of Eq. 2.15, the internal energy U can be written as

$$U = G\left(\varepsilon_{ij} - \frac{1}{3}\delta_{ij}\varepsilon_{kk}\right)^2 + \frac{1}{2}K\varepsilon_{kk}^2. \tag{2.17}$$

The quantity K is the bulk modulus, and G is the shear modulus.

The bulk and shear moduli are always positive and are related to the Lamé constants by [5, 32]

$$K = \lambda + \frac{2}{3}\mu \tag{2.18}$$

with $G = \mu$.

From Eq. 2.16, it is possible to write Hooke's Law. This law states that, for small deformations, the strain tensor is a linear function of the stress tensor, i.e. the deformation is proportional to the applied forces [5, 32]. Therefore, starting from Eq. 2.13

$$\begin{aligned}\sigma_{ij} &= \frac{\partial}{\partial\varepsilon_{ij}}\left(\frac{1}{2}\lambda\varepsilon_{ii}^2 + \mu\varepsilon_{ij}^2\right) = 2\mu\varepsilon_{ij} + \lambda\delta_{ij}\varepsilon_{kk} \\ &= K\varepsilon_{kk}\delta_{ij} + 2G\left(\varepsilon_{ij} - \frac{1}{3}\varepsilon_{kk}\delta_{ij}\right),\end{aligned} \tag{2.19}$$

[2] A solid material is said to be elastically isotropic when its elastic properties are the same in all directions [32].

and, by taking the trace of σ_{ij}, it is possible to write

$$\begin{aligned} \mathrm{tr}(\sigma_{ij}) = \sigma_{ii} = 3K\varepsilon_{ii} + 0 \\ \Leftrightarrow \varepsilon_{ii} = \frac{1}{3}\sigma_{ii}. \end{aligned} \tag{2.20}$$

Then, Eq. 2.19 can be inverted as:

$$\varepsilon_{ij} = \frac{1}{9K}\delta_{ij}\sigma_{kk} + \frac{1}{2G}\left(\sigma_{ij} - \frac{1}{3}\delta_{ij}\sigma kk\right) \tag{2.21}$$

The previous equation presents the linear dependence between the strain and stress tensors stated by Hooke's Law.

2.3.1 Young's modulus and Poisson's ratio

Apart from the bulk and shear moduli, there are two other useful quantities that can be derived. To do so, consider a simple case of homogeneous deformation of a rod. In a homogeneous deformation, the strain tensor ε_{ij} is constant throughout the body, which means that the stress tensor σ_{ij} is also constant [5]. Due to the nature of the deformation, the stress tensor can be determined from the boundary condition

$$\sigma_{ij}n_j = F_i, \tag{2.22}$$

where F_i are the external applied forces per unit area of the surface and n_j is the unit vector perpendicular to the surface. Since there are no external forces applied on the sides of the rod, on these sides Eq. 2.22 becomes

$$\sigma_{ij}n_j = 0. \tag{2.23}$$

Consider that the rod is aligned along the x_3-axis and only feels a force at its ends. The vector $\mathbf{n}$ of the side surface is perpendicular to the x_3-axis, i.e. $n_3 = 0$. The only non-zero component of σ_{ij} is, then, σ_{33}. On the end surface the condition is $\sigma_{33} = F$.

From Eq. 2.21, all the components σ_{ij} with $i \neq j$ are zero. The only non-zero components are [5]

$$\varepsilon_{11} = \varepsilon_{22} = -\frac{1}{3}\left(\frac{1}{2G} - \frac{1}{3K}\right)F \text{ and } \varepsilon_{33} = \frac{1}{3}\left(\frac{1}{3K} + \frac{1}{G}\right)F. \tag{2.24}$$

The component ε_{33} gives the relative elongation of the rod and, from it, it is possible to define the modulus of extension or Young's modulus E [5, 32],

$$\begin{aligned} \varepsilon_{33} = \frac{F}{E} &\Leftrightarrow E = \frac{F}{\varepsilon_{33}} \\ &\Leftrightarrow E = \frac{9KG}{(3K+G)}. \end{aligned} \tag{2.25}$$

The components ε_{11} and ε_{22} give the relative compression of the rod in the transverse direction. The ratio between the transverse compression and the longitudinal extension gives the Poisson's modulus ν [5, 32],

$$\begin{aligned} \varepsilon_{11} = -\nu\varepsilon_{33} &\Leftrightarrow \nu = -\frac{\varepsilon_{11}}{\varepsilon_{33}} \\ &\Leftrightarrow \nu = \frac{3K-2G}{2(3K+G)}. \end{aligned} \tag{2.26}$$

As said before, the values of K and G are always positive. Therefore, Poisson's ratio ν can have values in the interval

$$-1 \leq \nu \leq \frac{1}{2} \tag{2.27}$$

With these two new quantities defined, it is possible to rewrite the stress tensor σ_{ij} as

$$\sigma_{ij} = \frac{E}{1+\nu}\left(\varepsilon_{ij} + \frac{\nu}{1-2\nu}\varepsilon_{kk}\delta_{ij}\right) \tag{2.28}$$

and the strain tensor ε_{ij} as

$$\varepsilon_{ij} = \frac{1}{E}\left[(1+\nu)\sigma_{ij} - \nu\sigma_{kk}\delta_{ij}\right]. \tag{2.29}$$

2.4 Elastic properties of crystals

Consider the compression of a crystal where the effects of temperature are neglected. The change in the internal energy, like in isotropic bodies, is a quadratic function of the strain tensor ε_{ij}. As shown in the previous section, for an isotropic body this function contains just two coefficients. For crystals, though, the number of coefficients can be much larger [5]. The internal energy U of a deformed crystal can be written as

$$U = \frac{1}{2}\lambda_{ijkl}\varepsilon_{ij}\varepsilon_{kl}, \tag{2.30}$$

where the rank four tensor λ_{ijkl} is called the elastic moduli tensor. This tensor can be constructed to have the same symmetry properties as the strain tensor. Therefore, from symmetry

$$\lambda_{ijkl} = \lambda_{jikl} = \lambda_{ijlk} = \lambda_{klij}. \tag{2.31}$$

In general, this type of tensor has 81 components but only 21 are independent. The independent components arise due to the symmetry properties imposed by construction [5, 33]. This number of components can be reduced if the crystal possesses symmetry. The crystals' symmetries create relations between the various components of the tensor. The number of independent parameters N of the tensor σ_{ij} for the classes of various systems is presented in Table 2.1. These independent parameters could be elastic moduli or angles defining the orientation of axes in the structure [5].

As done in Section 2.3, it is still possible to write the relation between the stress tensor and the strain tensor. Using Eq. 2.30 the relation can be written as

$$\sigma_{ij} = \frac{\partial U}{\partial \varepsilon_{ij}} = \lambda_{ijkl}\varepsilon_{kl}. \tag{2.32}$$

Table 2.1: Number of independent parameters N of the tensor σ_{ij} for the classes of various systems. These independent parameters could be elastic moduli or angles defining the orientation of axes in the structure [5].

Class of the system	N	Class of the system	N
Triclinic	21	Rhombohedral (C_3, S_6)	7
Monoclinic	13	Rhombohedral (C_{3v}, D_3, D_{3d})	6
Orthorhombic	9	Hexagonal	5
Tetragonal (C_4, S_4, C_{4h})	7	Cubic	3
Tetragonal (C_{4v}, D_{2d}, D_4, D_{4h})	6		

2.4.1 Polycrystal

The discussions presented in Sections 2.3 and 2.4 are only valid for single crystals. Assuming that the components of a polycrystal are small enough, this can be seen as an isotropic crystal. Therefore, a polycrystal has only two moduli of elasticity, and it is necessary to know their expressions. Since, in general, there is no general relation between the moduli of elasticity of a polycrystal and those of a single crystal of the same substance, it is necessary to find a new way to compute them [5].

To do so, it is useful to use the Voigt notation. With this notation the components of the symmetric tensor λ_{ijkl} can be written in a compact way. Since the indices i, j, k, l are Cartesian indices, they take the values x, y and z. Then, under the transformations

$$xx \mapsto 1,\ yy \mapsto 2,\ zz \mapsto 3,\ yz = zy \mapsto 4,\ xz = zx \mapsto 5,\ xy = yx \mapsto 6, \tag{2.33}$$

the tensor components can be written as $\lambda_{xxxx} \mapsto \lambda_{11}$, $\lambda_{xxyy} \mapsto \lambda_{12}$ and so on. There are three ways to compute the bulk and shear moduli of a polycrystal. The first, the Voigt average, gives the upper bound of the moduli [4]

$$K_{\mathrm{V}} = \frac{1}{9}\big[(\lambda_{11} + \lambda_{22} + \lambda_{33}) + 2(\lambda_{12} + \lambda_{23} + \lambda_{31})\big] \tag{2.34}$$

$$G_{\mathrm{V}} = \frac{1}{15}\big[(\lambda_{11} + \lambda_{22} + \lambda_{33}) - (\lambda_{12} + \lambda_{23} + \lambda_{31}) + 3(\lambda_{44} + \lambda_{55} + \lambda_{66})\big]. \tag{2.35}$$

The second, the Reuss average, gives the lower bound of the moduli [4]

$$\frac{1}{K_{\mathrm{R}}} = (\lambda_{11}^{-1} + \lambda_{22}^{-1} + \lambda_{33}^{-1}) + 2(\lambda_{12}^{-1} + \lambda_{23}^{-1} + \lambda_{31}^{-1}) \tag{2.36}$$

$$\frac{15}{K_{\mathrm{R}}} = 4(\lambda_{11}^{-1} + \lambda_{22}^{-1} + \lambda_{33}^{-1}) - 4(\lambda_{12}^{-1} + \lambda_{23}^{-1} + \lambda_{31}^{-1}) + 3(\lambda_{44}^{-1} + \lambda_{55}^{-1} + \lambda_{66}^{-1}). \tag{2.37}$$

Finally, the Voigt-Reuss-Hill (VRH) average is the average of the upper and lower bounds of the moduli [4]

$$K_{\mathrm{VRH}} = \frac{1}{2}(K_{\mathrm{V}} + K_{\mathrm{R}}) \tag{2.38}$$

$$G_{\mathrm{VRH}} = \frac{1}{2}(G_{\mathrm{V}} + G_{\mathrm{R}}). \tag{2.39}$$

2.5 A Vickers' hardness model

The hardness of a material is its resistance to local deformation induced by pressing a harder solid [6]. Despite the existence of many scales to measure the hardness of a material, in this thesis, only Vickers' hardness will be considered.

Vickers' hardness can be measured experimentally by applying a square-based pyramid diamond load into a sample and measuring the resulting area of indentation [6]. Then, after measuring the area of indentation, the Vickers' hardness H_{v} can be determined as a function of the force F applied to the diamond, the surface area A of the indentation using [6]

$$H_\mathrm{v} = \frac{F}{A}, \tag{2.40}$$

or, alternatively, as a function of the force F applied to the diamond and the average length d of the diagonal left by the indenter using [6]

$$H_\mathrm{v} \approx 1.8544\frac{F}{d^2}. \tag{2.41}$$

The experimental measurement of the hardness values can be difficult since the applied stress depends on various factors, such as the orientation of the material, the loading forces, and the geometry of an indenter [6]. Therefore, the development of models that can easily estimate this property is crucial. Most of these models have free parameters that are adjusted to reproduce experimental data. These models can be divided into two types [6]:

- Type 1: These models try to relate other elastic properties such as the bulk and shear moduli, since they are easier to measure and a lot of experimental data is available, with the hardness H_v of a material. One of these models, the Chen *et al.* model [34], relates the bulk modulus K and the shear modulus G to the hardness as

  $$H_\mathrm{v} = 2\left(\frac{G^3}{K^2}\right) - 3. \tag{2.42}$$

  By making use of the homogeneous approximation

  $$K = \frac{E}{3(1-2\nu)}; \quad G = \frac{E}{2(1+\nu)}, \tag{2.43}$$

  Eq. 2.42 can be rewritten in terms of Young's modulus E and Poisson's ratio ν as [6]

  $$H_\mathrm{v} = 2\left(\frac{9E(1-2\nu)^2}{8(1+\nu)^3}\right)^{0.585} - 3. \tag{2.44}$$

  Despite offering good results for a wide variety of materials, the Chen *et al.* model [34] has some limitations [6]:

  – overestimates the hardness of materials that have low or negative Poisson's ratio;

  – incorrectly predicts the hardness for unusually hard materials;

  – give unphysical negative hardness values for soft compounds.

- Type 2: These models attempt to calculate the hardness from the chemical bond properties [6]. In Table 2.2 are shown examples of this type of models as well as the properties used to calculate the hardness.

Table 2.2: Information about models that attempt to calculate the hardness from the chemical bond properties [6].

Model	Chemical bond properties
Gao's model [35]	Bond length Average electron density Ionicity
Šimůnek and Vackář's model [36]	Chemical bond strength
Li *et al.*'s model [37]	Electronegativities Covalent radii Bond length
Lyahov-Oganov's model [38]	Bond-valence model

In this thesis, the model used to calculate Vickers's hardness falls into the previous definition of a type 1 model and was proposed by Mazhnik and Oganov in [6]. This model is inspired in Chen *et al.*'s model [34] and assumes that the hardness H_v can be written in terms of a dimensionless constant γ_0 independent of the material, a dimensionless function of Poisson's ratio $\chi(\nu)$, and Young's modulus as

$$H_\mathrm{v} = \gamma_0 \chi(\nu) E. \tag{2.45}$$

From [6], the value of γ_0 is 0.096 and the function $\chi(\nu)$ is

$$\chi(\nu) = \frac{1 - 8.5\nu + 19.5\nu^2}{1 - 7.5\nu + 12.2\nu^2 + 19.6\nu^3}. \tag{2.46}$$

According to mean absolute error (MAE) and root mean squared error (RMSE) values presented in Table 2.3, the Mazhnik and Oganov's model [6] used to calculate Vickers' hardness in this thesis shows good accuracy when compared with other models (see Table 2.3). From the informations in [6] this model also seems to solve the problems associated with Chen *et al.*'s model.

Table 2.3: Summary of the accuracy of some models used to calculate Vickers' hardness. The mean absolute error (MAE) and root mean squared error (RMSE) are provided for comparison [6].

Model	MAE [GPa]	RMSE [$\sqrt{\text{GPa}}$]
Gao's model [35]	3	3
Šimůnek and Vackář's model [36]	3	2
Chen *et al.*'s model [34]	3	4
Mazhnik and Oganov's model [6]	3	4

Chapter 3

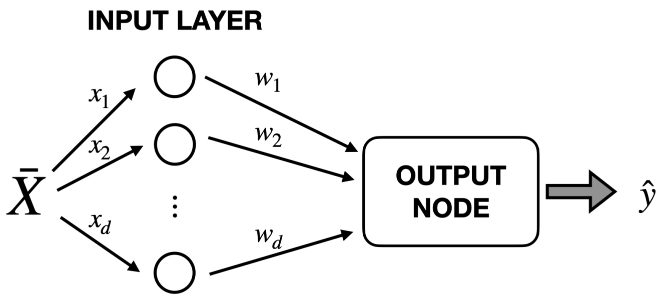
On neural networks

3.1 An overview

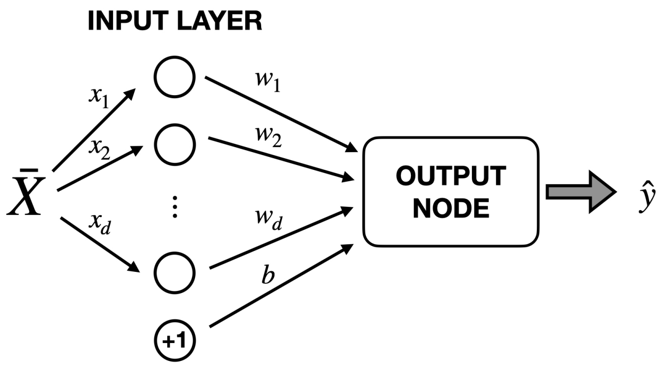
One of the most recognizable machine learning methods are artificial neural networks or simply neural networks (NN), having earned this name since they take inspiration from the learning mechanism in biological organisms. NNs are composed by layers of nodes, also known as neurons, connected through weights [13]. A node is a basic computational element of a NN that: (i) receives a scalar from the input data or from the linear combination of the outputs of all previous nodes, (ii) applies a non-linear function to the scalar whose result will be used as the output of node. The first layer of a NN is called input layer. The nodes in this layer do not perform any computational work and only transmit the input data to the next layer. The nodes in the remaining layers of the NN are responsible for all the computations involved in a NN. These layers are called computational layers [13].

The simplest NN its called perceptron and contains a single input layer and a output node (see Figure 3.1a). The inputs of a node are scaled with weights w that are used as intermediate values to compute the output of that node. These output values are then propagated through the NN until reaching the output node [11, 13].

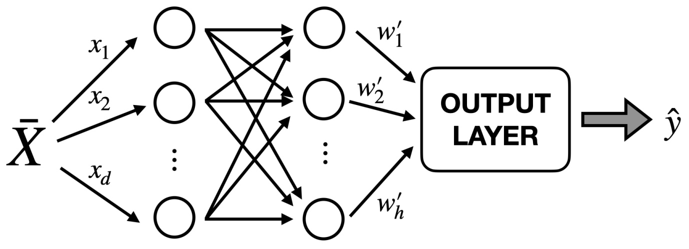
The learning process occurs by changing the weights that connect the nodes using training data containing examples of input-output pairs $(\bar{X}, y)$ of the function to be learned [11, 13]. The NN is fed with the inputs $\bar{X}$ to make predictions $\hat{y}$ about the outputs y. The value of $\hat{y}$ is then used to evaluate the accuracy of the NN by comparing it with the original output label y [13]. The weights are adjusted successively over many input-output pairs to refine the function computed by the NN to improve prediction accuracy. This refinement is called training the network. When the training is complete, the NN will be able to perform good predictions over unseen inputs, this being called model generalization [11, 13, 39].



(a) Single layer neural network.



(b) Single layer neural network with a bias node.



(c) Multi-layer neural network.

Figure 3.1: Examples of neural networks.

3.2 Multi-layer neural network

Multi-layer NN are networks that contain more than one computational layer. Consider a simple case of a multi-layer NN containing one input layer, one intermediate layer, and one output layer (see Figure 3.1c). The input layer with d nodes transmits the features $\bar{X} = [x_1 \cdots x_d]$ with edges of weights $\bar{W} = [w_1 \cdots w_d]$ to the intermediate layer with h nodes. Since the computations performed by the intermediate layers are not visible to the user, they are also called hidden layers. The hidden layer transmit the h features $\bar{X}' = [x'_1 \cdots x'_h]$ with edges of weights $\bar{W}' = [w'_1 \cdots w'_h]$ to an output layer with c nodes.

A multi-layer NN is also referred to as a *feed-forward* network, since successive layers feed into one another in the forward direction. The mathematical explanation of these networks follows what is done in [40] and is presented in Sections 3.2.1 and 3.2.2.

3.2.1 *Forward* Phase

Consider a node j in one of the h hidden layers which is connected to the nodes of the previous layer by weights w_{ji}. The total input of the node j is a linear function of the outputs x_i of the nodes that are connected to him and the weights w_{ji} of those connections

$$\varSigma = \bar{W} \cdot \bar{X} = \sum_i x_i w_{ji}. \tag{3.1}$$

The computations performed in the node j can be divided into two different phases, as shown in Figure 3.2. In the first phase, called the pre-activation phase, the node computes the Eq. 3.1 which is called the pre-activation value a. The other phase computes the output value y_j of the node, which is called the post-activation value h, by applying to x_j a activation function [13]. The choice of the activation function is important in the design of NN. Depending on the target variable to predict, non-linear functions such as the *sign*, *sigmoid*, or *hyperbolic tangents* may be used as activation functions of various layers [13]. Therefore, choosing a generic activation function σ, the generic output of a node is given by

$$\hat{y} = \sigma(\bar{W} \cdot \bar{X}) \tag{3.2}$$

For some applications, the predictions have an invariant part called bias. This bias can be added to the network by considering a bias node with weight b that always transmits a value 1 to the nodes to which it is connected (see Figure 3.1b). In this case, Eq. 3.2 should be written as

$$\hat{y} = \sigma(\bar{W} \cdot \bar{X} + b), \tag{3.3}$$

where b is the bias. For simplicity, in the following discussions, bias nodes will not be considered.

This is the end of the *forward* phase, in which the training instances that where fed to the network are forward propagated through the network.

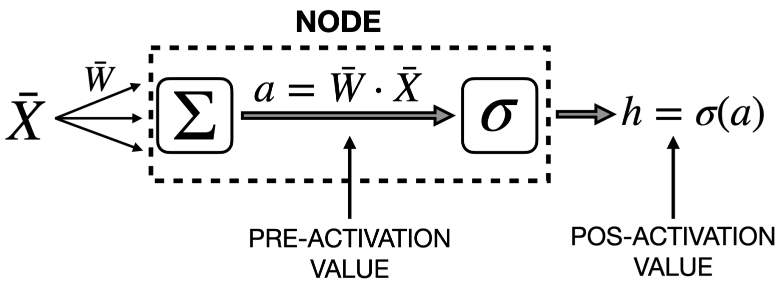


Figure 3.2: Representation of a computational node and its calculations.

3.2.2 *Backwards* Phase: the back-propagation

The goal is now to find a set of weights that minimise the error $E(\bar{X})$ between the output value $\hat{y}$ predicted by the network and the expected output value y given to the network [13, 40],

$$E(\bar{X}) = \hat{y} - y. \tag{3.4}$$

If this error $E(\bar{X})$ is non-zero, then the weights need to be updated by minimizing an objective function $\mathcal{L}$, called loss function, that depends on the prediction error E in the dataset $\mathcal{D}$

$$\text{Minimize}_{\bar{W}} \mathcal{L} = \sum_{(\bar{X},y) \in \mathcal{D}} \mathcal{L}(y - \hat{y}). \tag{3.5}$$

In practice, the error $E(\bar{X})$ does not reach zero but eventually converges to a small value.

The most common way to perform the minimization shown in Eq. 3.5 is to use gradient descent [11, 13, 40]. The gradient descent method minimizes the function $\mathcal{L}(w)$ by updating the parameters w in the opposite direction of the gradient of $\mathcal{L}(w)$ with respect to w that has largest absolute value [41]. The gradient descent is given by

$$w_{\text{new}} = w - \alpha \boldsymbol{\nabla}_w \mathcal{L}(w), \tag{3.6}$$

with the learning rate α being the size of the steps taken. After the forward phase, a back pass, also known as back propagation, is applied to propagate the partial derivatives of $\mathcal{L}$ concerning each weight throughout the network [40]. The back-propagation starts by computing $\frac{\partial \mathcal{L}}{\partial y}$ for each output node and, by applying the chain rule, also computes

$$\frac{\partial \mathcal{L}}{\partial x_j} = \frac{\partial \mathcal{L}}{\partial y_j} \cdot \frac{\mathrm{d} y_j}{\mathrm{d} x_j} = \frac{\partial \mathcal{L}}{\partial y_j} \cdot \frac{\mathrm{d}\sigma(x_j)}{\mathrm{d} x_j}. \tag{3.7}$$

From Eq. 3.7 is possible to know how a change in the input x_j of a output node j affects the error. The definition of this input x_j given in Eq. 3.1 makes it possible to compare how the error will be affected by the change of these states x and weights w [40]. For weights the process is similar. Assuming a weight w_{ji}, from i to j the derivative is [40]

$$\frac{\partial \mathcal{L}}{\partial w_{ji}} = \frac{\partial \mathcal{L}}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}} = \frac{\partial \mathcal{L}}{\partial x_j} \cdot y_i. \tag{3.8}$$

Then, the general formula to update the weights of the network can be obtained by rewritten Eq. 3.6 as

$$w_{\text{new}} = w - \alpha \frac{\partial \mathcal{L}}{\partial w_{ji}} \tag{3.9}$$

$$= w - \alpha \frac{\partial \mathcal{L}}{\partial x_j} \cdot y_i. \tag{3.10}$$

3.2.3 Gradient Descent: variants and optimizations

As previously stated, the update of the weights is responsible for the learning process of the network. Therefore, it is important to understand how the amount of data used to compute the gradients affects its performance [42].

Consider a dataset with N input-output pairs $(\bar{X}, y)$ and let a batch be a portion with $n < N$ elements of that dataset. The gradient of the loss function $\mathcal{L}$ with respect to the weights w can be computed using the entire dataset as

$$w_{\text{new}} = w - \alpha \boldsymbol{\nabla}_w \mathcal{L}(w), \tag{3.11}$$

i.e, the weights are updated just once. However, it is impractical to simultaneously run all examples through the network to compute the gradient with respect to the entire dataset in one shot, since depending on the size of the dataset, it would be very slow and computationally heavy [13, 42].

Other way of calculate the gradient is to update the weights for each pair $(\bar{X}^{(i)}, y^{(i)})$,

$$w_{\text{new}} = w - \alpha \boldsymbol{\nabla}_w \mathcal{L}(w, \bar{X}^{(i)}, y^{(i)}). \tag{3.12}$$

This method is called stochastic gradient descent (SGD) and is faster than the previous method. However, performing frequent updates with a high variance causes the loss function $\mathcal{L}$ to fluctuate heavily. This fluctuation enables it to jump to other and potentially better local minima but also complicates convergence to the exact minimum [42].

In mini-batch stochastic gradient descent, the weights update is done using a batch with size $n < N$,

$$w_{\text{new}} = w - \alpha \boldsymbol{\nabla}_w \mathcal{L}(w, \bar{X}^{(i+n)}, y^{(i+n)}). \tag{3.13}$$

This method often provides the best trade-off between stability, speed, and memory requirements [13]. It is common to use powers of 2 as the size of the mini-batch and the usual batch sizes are 32, 64, 128, or 256 [13].

However, even the mini-batch stochastic gradient descent does not guarantee good convergence [13, 42]. To solve the convergence problem, it is possible to use gradient descent optimization methods like momentum [43] that helps to accelerate the SGD in the relevant directions, or methods like RMSprop [39] and Adam [44] that adaptively adjust the learning rate. For the rest of this thesis, the mini-batch stochastic gradient descent will be called stochastic gradient descent (SGD).

3.3 Convolutional Neural Networks

Convolutional neural networks (CNN) are a different type of network where the states in each layer are arranged according to a spatial grid structure. The spatial relations are inherited from one layer to the next and are a consequence of each feature value being based on a local spatial region in the previous layer [13]. The input $I(x, y)$ of a CNN has a grid-like structure and the pairs (x, y) represent the coordinates of a element in the grid. Usually, a CNN is built using a repetition of [13]:

- Convolution layers: A convolution layer contains a set of convolutional kernels, where each node acts as a kernel. The convolutions processes inside this layer divide the grid into small slices by convolving the kernel with the input I using a set of weights [45]. Assuming a $j \times k$ kernel $e(j, k)$ a convolution operation in layer l can be express in terms of the linear combination of the individual elements s of the input with the elements of the kernel as [45, 46]

$$f_l(p, q) = \sum_{c} \sum_{x,y} s(x, y) \cdot e(j, k). \tag{3.14}$$

  If the CNN has more than one kernel k, then Eq. 3.14 becomes

$$f_l^k(p, q) = \sum_{c} \sum_{x,y} s(x, y) \cdot e^k(j, k). \tag{3.15}$$

  An example of this operation using a 5×5 input and a 3×3 kernel is shown in Figure 3.4a.

- Pooling layers: Once the features are extracted using convolution layers, their exact location become less important as long as their position in relation to others is preserved [45]. Pooling or down-sampling offers a way to reduce the feature map by introducing a translation invariance to small shifts and distortions. The size reduction of the feature-maps to invariant feature set helps to keep the complexity of the model low and also to increase generalization by reducing overfit [13, 45, 46]. A generic pooled feature map $\mathbf{P}$ can obtained by applying a pooling operation ϑ to the feature-map $\mathbf{C}$,

$$\mathbf{P} = \vartheta(\mathbf{C}). \tag{3.16}$$

  In Figure 3.4b is shown an example of a 2×2 max pooling operation with a 2 dimensions input. This operation extracts slices from the input feature map and outputs the highest value in each slice. For example, if a 2×2 max pooling operation is done, the feature map will be downsample by a factor of 2. Another common pooling operation is the global average pooling. This operation performs the average of all values in the feature map and outputs its average [47]. Usually, the pooling layers are used before a multi-layer NN.

In summary, the repetition of convolution and pooling layers perform the extraction of the relevant input features that are then passed to an multi-layer NN similar to the ones presented in Section 3.2 that gives the network's output [13, 46, 47].

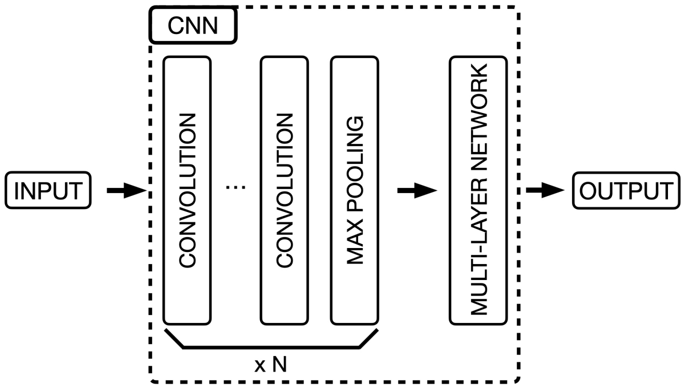


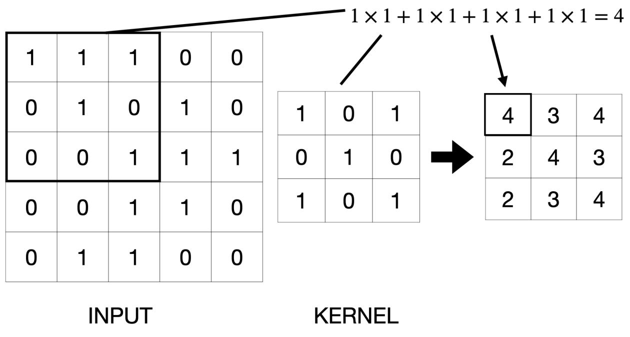
Figure 3.3: Representation of a generic CNN with N blocks of convolution and pooling layers. The network can have as much convolution layers before a pooling layer as needed for the task in hand. The multi-layer network can be substituted for a single-layer network depending on the task.

3.4 Hyperparameters

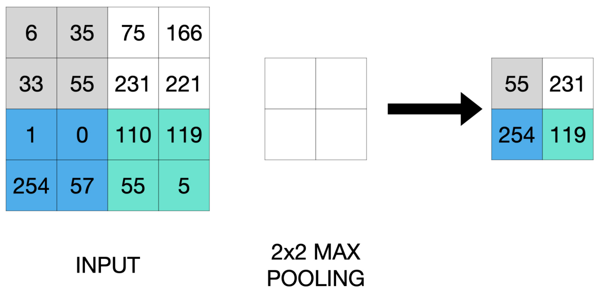
A good choice of parameters that define a network is a fundamental part of building that network. Looking at the networks described in Sections 3.2 and 3.3, the learning rate α, the number of hidden layers, the number of convolution and pooling layers, the dimension of the network's input, and the choice of the gradient-descent method can be seen as hyperparameters of a network. The hyperparameters λ are used to configure various aspects of the model and can have wildly varying effects on the resulting model and its performance [48]. Usually, the values of the hyperparameters are not adapted by the learning algorithm itself [11].

The set of hyperparameters λ is also related to the complexity of the model. If the model is too complex, it will tend to overfit, i.e, the fit to the training data will be very good but will fail to generalise for unseen data. On the other hand, if the model is too simple, it will tend to underfit, i.e it won't capture all the information in the data [48].

The parameters that control the capacity of the model should always be hyperparameters. If those hyperparameters were to be learned on the training set, they would always choose the maximum possible model capacity. This choice would make the model overfit the data [11]. A way of solving this problem is to use a validation set with



(a) Example of the convolution of a 5 × 5 input with a 3 × 3 kernel.



(b) Example a 2 × 2 max pooling of a 4 × 4 input.

Figure 3.4: Examples of application of convolution and pooling. In Figure (a) a 3 × 3 kernel is applied to a 5 × 5 input to demonstrate the feature extraction using a convolutional layer. In Figure (b) a 2 × 2 max pooling operation is applied to a 4 × 4 input to demonstrate the feature extraction using a pooling layer.

data that the training process has not seen to choose the best set of hyperparameters. This method of using a validation set constructed from the same distribution of the training set is called cross-validation [11].

In cross-validation, the training set is used to learn the model's parameters and the validation set is used to estimate the generalization error during or after training. The generalization error allows for the hyperparameters to be updated accordingly [11].

There are multiple ways to find the best set of hyperparameters. The simplest way is to monitor the network and manually tune the hyperparameters. In addition to being time consuming, this method involves a guessing factor which makes it difficult to apply to complex models. Another method called gridsearch [49, 50], used in this thesis, consists of choosing multiple values for each hyperparameter, run every possible combination of them, and then choose the combination that provides the best results. The optimization methods for the SGD can also be used to search for the best learning rate. This is, the learning rate at which the learning process ends can be used as the optimal learning rate [41, 50]. Other hyperparameter search methods can be found in [50].

After the best set of hyperparameters is found, the generalization error may be estimated using a test set that should not contain data from the training and validation sets [11]. In this work, 60% of the full database was used as the training set, 20% as the validation set, and the remaining 20% as the test set.

Chapter 4

On the model

4.1 Crystal graph convolution neural network

The model used in this thesis was the Crystal Graph Convolutional Neural Network (CGCNN) presented by Xie and Grossman in [2]. In this model, the crystal structure is represented by a crystal graph that encodes both atomic information and bond interactions between atoms. In a crystal graph $\mathcal{G}$, as the one in Figure 4.1, the nodes represent atoms, and the edges represent the connections between atoms in a crystal. The information of the atom corresponding to the node i is encoded by a feature vector $\mathbf{v}_i$ and each node in the crystal graph is connected to its 12 nearest neighbours. The number of nearest neighbours to which each node is connected comes from practice. The k-th bond connecting atoms i and j is represented by a feature vector $\mathbf{u}_{(i,j)_k}$ [2, 3]. More details about the atom and bond feature vectors are presented in Section 4.2.

On top of the crystal graph $\mathcal{G}$, a convolutional neural network is built to automatically extract representations that are optimal for predicting target properties by training with data [2, 3].

4.1.1 Model architecture

The convolutional neural networks built on top of the crystal graph are used to automatically extract representations that are good for predicting target properties, and are similar to the ones described in Section 3.3. Before the convolution layers are used to iteratively update the atom feature vector $\mathbf{v}_i$, a neighbour feature vector

$$\mathbf{z}^{(t)}_{(i,j)_k} = \mathbf{v}^{(t)}_i \oplus \mathbf{v}^{(t)}_j \oplus \mathbf{u}_{(i,j)_k}, \tag{4.1}$$

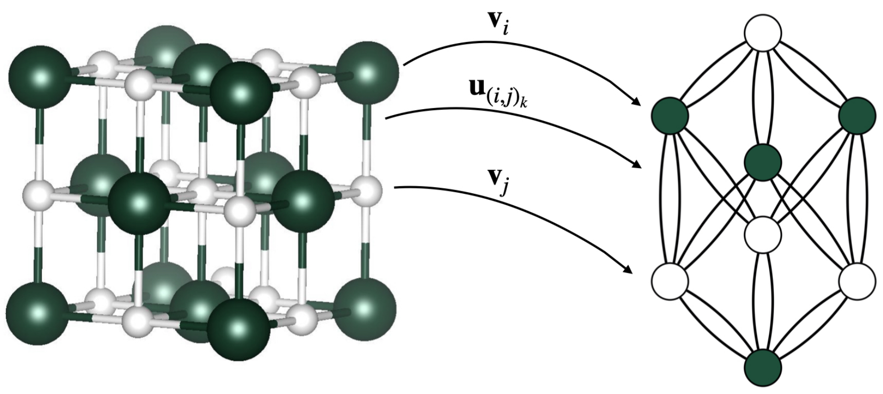


Figure 4.1: Representation of a crystal graph $\mathcal{G}$. The crystal is converted to a crystal graph where the nodes represent atoms in the unit cell and edges represent the atom connections. Both nodes and edges are characterized by vectors corresponding to the atoms and bonds in the crystal, respectively [2, 3].

is created by concatenating[1] the atom and bond feature vectors of the neighbouring atoms of the i-th atom [2, 3]. Then, the update of the atom feature vector $\mathbf{v}_i^{(t)}$ can be written as

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t)} + \sum_{j,k} \sigma\left(z_{(i,j)_k}^{(t)} \mathbf{W}_f^{(t)} + \mathbf{b}_f^{(t)}\right) \odot g\left(\mathbf{z}_{(i,j)_k}^{(t)} \mathbf{W}_s^{(t)} + \mathbf{b}_s^{(t)}\right), \tag{4.2}$$

where $\odot$ denotes element-wise multiplication, σ denotes a sigmoid function, and g is the activation function for introducing non-linear coupling between layers. The quantities $\mathbf{W}_s^{(t)}$ and $\mathbf{W}_f^{(t)}$ are the self weight matrix, and weight matrix of the t-th layer, respectively [2, 3].

After R convolutions, a new feature vector $\mathbf{v}_i^{(R)}$ is generated for each atom i of the crystal. The pooling layer after the R convolution layers produce the overall feature vector $\mathbf{v}_c$ for the crystal

$$\mathbf{v}_{\boldsymbol{c}} = \text{Pool}\left(\mathbf{v}_0^{(R)}, \mathbf{v}_1^{(R)}, \ldots, \mathbf{v}_N^{(R)}\right), \tag{4.3}$$

using the atom feature vectors from the N atoms of the crystal.

[1] The concatenation operation in Eq. 4.1 is represented by the $\oplus$ symbol.

The overall feature vector $\mathbf{v}_c$ for the crystal is then used as input to a multi-layer NN with L_2 hidden layers, as the one presented in Section 3.2, to predict a target property value y. The predicted value can be obtained by using Eq. 3.2 as

$$\hat{y} = f(\mathbf{v}_c \mathbf{W}_c + \mathbf{b}_c), \tag{4.4}$$

where $\mathbf{W}_c$, $\mathbf{b}_c$, and $f(\cdot)$ are the weight matrix, bias, and non-linear activation function of the layers that compose the multi-layer NN, respectively [2, 3, 51]. The training process is done as described in Section 3.2 using backpropagation and the SGD to update the weights. A schematic representation of the model is shown in Figure 4.2.

Depending on the activation function of the output layer and the loss function used, the output of the multi-layer NN could be a prediction task or a classification task. The loss function and the activation function of the output layer for the regression used in each task are presented in Table 4.1.

Table 4.1: Loss function and the activation function of the output layer for the regression task used in this thesis and the functions for the classification task.

Type	Loss function	Output layer activation function
Regression	*Mean squared error* (MSE)	*Linear*
Classification	*Negative log-likelihood* (NLL)	*LogSoftmax* or *LogSigmoid*

4.1.2 Improvements on the original CGCNN model

The CGCNN model used in this work was firstly published in 2018 in [2] and, since then, new models based on it have been developed. This list might not be exhaustive, but it shows the main improvements:

- *MT-CGCNN*: This was the first altered version of the CGCNN model published in 2018 [52]. This model is in every way similar to the original one, the only difference being the inclusion of a hard parameter sharing multi-task learning[2], where for each crystal property (p) being learned, there is an independent multi-layer network which takes $\mathbf{v}_c$ and predicts the intended property value. This

[2] A common set of layers is shared across all tasks and then some task-specific output layers are used for each individual task.

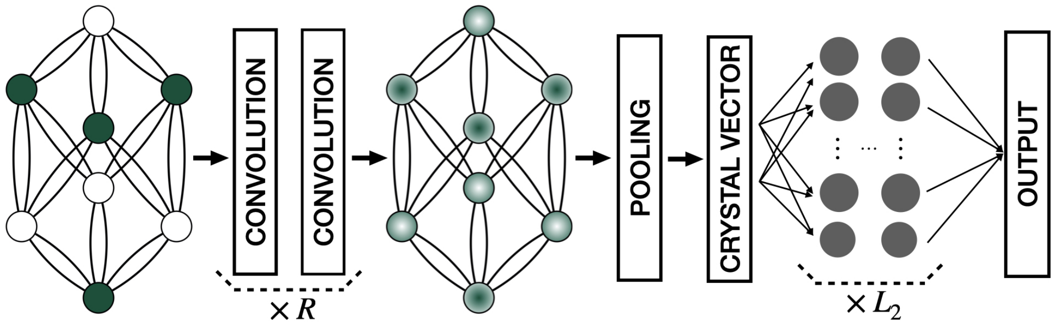


Figure 4.2: Representation of the CGCNN model. The result of the R convolutional layers is a new graph with each node representing the local environment of each atom. After pooling, a vector $\mathbf{v}_c$ representing the entire crystal is connected to a multi-layer NN with L_2 hidden layers, followed by the output layer to provide the prediction value [2, 3].

architecture allows to predict more than one property at the same time, while CGCNN only allows to predict one property at a time.

- *iCGCNN*: A second modification was published in 2020 [22]. In this architecture, descriptors extracted from the crystal graphs include the information of the Voronoi tessellated crystal structure, explicit three-body correlations of neighbouring constituent atoms, and an optimized chemical representation of interatomic bonds, all of which are absent in the crystal graphs utilized by the original CGCNN model. The goal of this architecture was to not only improve the accuracy of the original model, but also have a better representation of the chemical nature of an inorganic compound.

- *TL-CGCNN*: A third altered version was published in 2021 [51]. These architecture propose the use of transfer learning (TL) to improve the accuracy of the original model. The TL is used to provide knowledge to other predictive models with different target variables. Therefore, the focus of this architecture is to load a pre-trained model constructed with an enormous amount of data and use it to make predictions on a smaller database.

4.2 The feature vectors

The atom properties of the original model are encoded in atom feature vector $\mathbf{v}_i$ using one hot encoding. For discrete values, the vectors are encoded according to the category that the value belongs to; for continuous values, the range of property values is evenly divided in 10 categories and the vectors are encoded accordingly [2, 3, 53]. The full list of atom properties as well as their ranges are presented in Table 4.2. The atom feature vector constructed this way will be denoted as $\mathbf{v}_i^{(0)_{\text{orig}}}$.

Table 4.2: Properties used in the original atom feature vector $\mathbf{v}_i^{(0)_{\text{orig}}}$ [2, 3].

Property	Range	# of categories
Group number	$1, 2, \ldots, 18$	18
Period number	$1, 2, \ldots, 91$	9
Electronegativity	$0.5 - 4.0$	10
Covalent radius (pm)	$25 - 250$	10
Valence electrons	$1, 2, \ldots, 12$	12
First ionization energy (log($\mathbb{V}$))	$1.3 - 3.3$	10
Electron affinity (eV)	$-3 - 3.7$	10
Block	s, p, d, f	4
Atomic volume (log(cm^3/mol))	$1.5 - 4.3$	10

Apart from the original atom feature vector, a different one was used in this thesis, where only the atomic number is used (see Table 4.3). This means that the only information given to the network was that all atoms were different from each other. The atom feature vector constructed this way will be denoted as $\mathbf{v}_i^{(0)\text{z}}$.

Table 4.3: Properties used in the new atom feature vector $\mathbf{v}_i^{(0)\text{z}}$.

Property	Range	# of categories
Atomic number	$1, 2, \ldots, 100$	100

To build the bond feature vector $\mathbf{u}_{(i,j)}^{(0)}$, the distance between the connected atoms $d_{(i,j)}$ is expanded in a Gaussian basis as [3]

$$\mathbf{u}_{(i,j)}^{(0)}[t] = \exp\left(-\frac{\left(d_{(i,j)} - \mu_t\right)^2}{\sigma^2}\right) \tag{4.5}$$

where $\mu_t = 0.2t$ Å for $t = 0, 1, .., k$ and $\sigma = 0.2$ Å.

Table 4.4: Properties used in the bond feature vector $\mathbf{u}_{(i,j)}^{(0)}$ [3].

Property	Range	# of categories
Atom distance (Å)	0.7 – 5.2	10

4.3 The code

The original code of the model was written in the Machine Learning framework *Pytorch* [54]. Therefore, the first step was to rewrite it in a framework called *Pytorch Lightning* [30]. This decision was made due to *Pytorch Lightning's* simpler and user friendly interface. It is worth noting that the core of the original code [2, 3] was kept unchanged.

The change of framework made it possible to easily implement a Early Stop functionality which allows the training process to stop when a pre-defined metric, usually the loss function $\mathcal{L}$, stops improving. This functionality helped in the reduction of the computational time.

The ability of classification of the model was also improved. The original code only includes the possibility of binary classification. Therefore, a way to perform multi-class classification was implemented. Other implementation made was the ability to remove the values of the network before the last activation function. This implementation allows to use these values as input features in simpler models to explore the accuracy of transferring the feature engineering using a huge dataset, to small and specific datasets.

Finding the best model for each property implies finding the set of hyperparameters for which the accuracy is the best. The package *Optuna* [31] was used to help manage and perform the gridsearch of the hyperparameters. In addition to being able to automatically run all the possible combinations of hyperamaters chosen for the model (see Table 4.5), this package was chosen due to its routine `pruners.MedianPrune`. This routine considers each combination of hyperparameters as a trail and prunes the current trail if the trial's best intermediate result is worse than the median of intermediate results of previous trials at the same step. Therefore, the only weights that are saved are the ones from not pruned trails.

From all the sets of hyperparameters whose weights were saved, the best one is chosen according to the result of a certain pre-defined metric. For the regression task, the pre-defined metric used was the mean absolute error (MAE) between the predicted and target values from the validation step.

Table 4.5: Values of the hyperparameters used in the gridsearch to obtain each property best model.

Hyperparameter	Range
Number of convolutional layers R	$1, 2, \ldots, 6$
Number of hidden layer L_2	$1, 2, \ldots, 6$

Chapter 5

On the databases

5.1 The training database

The models for the shear G and bulk K moduli were trained using the [4] database. The raw data used to generate this database was the calculated properties from the Materials Project [9]. In contrast with the data from the Materials Project, the [4] database contain only the materials input objects and target variables. Apart from the removal of extra data, the following filters were also applied [4, 8]:

- Removal of entries having a energy above the convex hull[1] above 150 meV;
- Removal of entries having G_{V}, G_{R}, G_{VRH}, K_{V}, K_{R}, or K_{VRH} less than or equal to zero;
- Removal of entries failing $G_{\mathrm{R}} \leq G_{\mathrm{VRH}} \leq G_{\mathrm{V}}$ or $K_{\mathrm{R}} \leq K_{\mathrm{VRH}} \leq K_{\mathrm{V}}$;
- Removal of entries containing noble gases;
- Removal of all columns except structure and the target variable.

In total, the final database contains 9741 structures covering 87 elements. This database does not include materials containing Helium ($Z = 2$), Neon ($Z = 10$), Argon ($Z = 18$), Xenon ($Z = 54$), Polonium ($Z = 84$), Astatine ($Z = 85$), Radon ($Z = 86$), Francium($Z = 87$), and Radium ($Z = 88$).

The database created to train the models for Vickers' hardness H_{v} was built using the values of shear G and bulk K moduli contained in the previous database and Eqs. 2.45 and 2.46. Vickers' hardness database contained the same 9741 materials covering 87 elements as the other database.

[1]A discussion about the energy above the convex hull is presented in Appendix A

The plot of the number of materials of these databases that contain a certain chemical element is presented in Figure 5.1. The minimum $min(y)$, maximum $max(y)$, and mean $\bar{y}$ value as well as the standard deviation σ of each elastic property are presented in Table 5.1.

Table 5.1: Information about the the minimum $min(y)$, maximum $max(y)$, and mean $\bar{y}$ value as well as the standard deviation σ of the elastic properties used to train the models [4].

Property	min(y) [GPa]	max(y) [GPa]	$\bar{y}$ [GPa]	σ [GPa]
G	2	523	50	39
K	2	575	103	68
H_v	0	99	7	5

Since the models were trained using the decimal logarithm of the properties, the values used to plot the distribution of values of each property shown in Figure 5.2 were calculated using $\log\left(\frac{P}{P_0}\right)$, with P being the value of the elastic property in GPa and $P_0 = 1$ GPa. The reason behind the use of the decimal logarithm will be discussed in Section 6.1.

From the analysis of the Figure 5.1 it is possible to see that the database is biased. The number of materials with elements with $Z \leq 31$ is greater than the number of materials with elements with $Z > 31$. Oxygen ($Z = 8$) is present in 1635 materials and is the most frequent element in the materials in the database. The less frequent element is Neptunium ($Z = 93$) and is present in 19 materials.

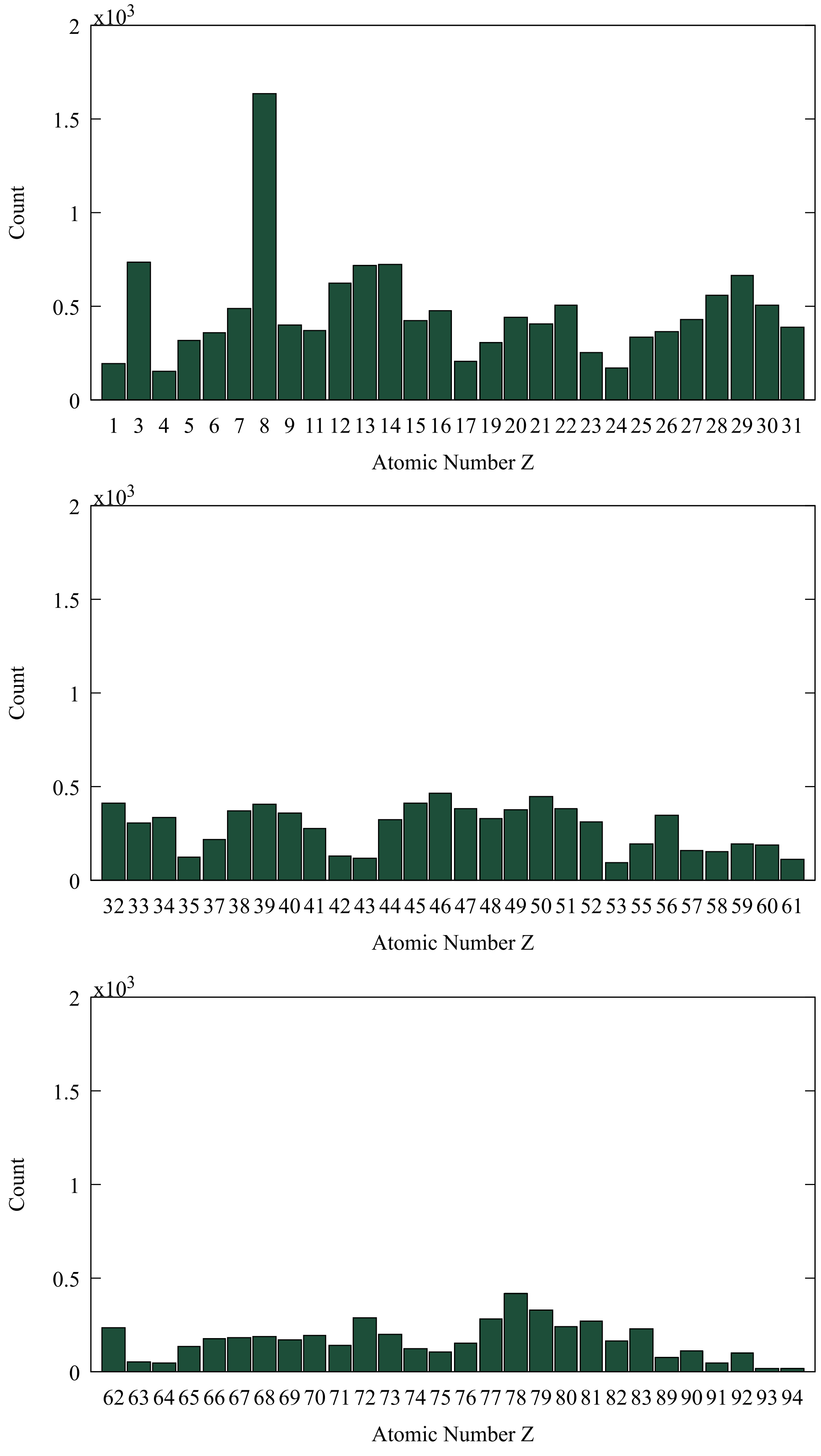


Figure 5.1: Representation of the number of materials containing a certain chemical element on the databases used to train the models.

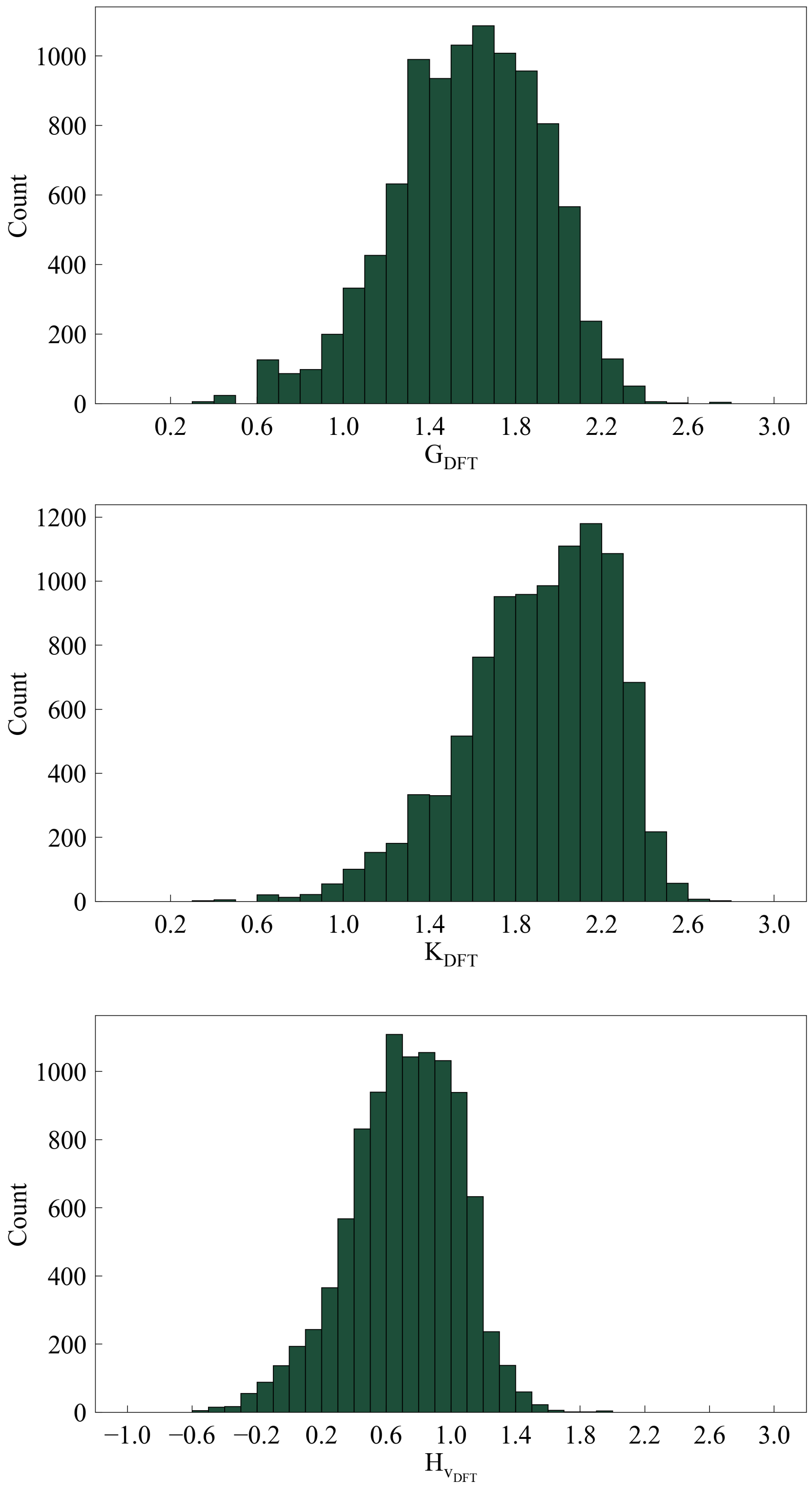


Figure 5.2: Distribution of values of the elastic properties shear G and bulk H moduli, and Vickers' hardness H_v of the materials contained in the databases used to train the models.

5.2 DB1M - a dataset form [1]

The models trained with the previous databases were to be used to predict the shear G and bulk K moduli and Vickers' hardness H_v of the materials in the database of [1] with the goal of finding super-hard materials. From now onwards this database will be called DB1B and the dataset used for training in [24] will be called DB2021.

The motivation for the creation of DB1B was to solve DB2021's bias with respect to the distribution of chemical elements and of crystal structures. As described in [1], the first bias problem was solved by performing a series of high-throughput searches with an extended set of 84 chemical elements, using the transfer learning approach presented in [24]. The additional data generated by these calculations was employed to retrain the crystal graph network (DCGAT-1) from [24]. The structural bias was reduced by using this DCGAT-1 to scan a material space of almost 1 billion compounds that comprises more than 2000 crystal-structure prototypes [1].

A dataset denoted as DB1M was built using 1160292 materials from DB1B with with energy above the convex-hull bellow 200 meV/atom. DB1M's materials did no have Polonium ($Z = 84$), Astatine ($Z = 85$), Radon ($Z = 86$), Francium ($Z = 87$), and Radium ($Z = 88$) in their constitution. In contrast with the databases discussed in Section 5.1, DB1M includes materials containing Helium ($Z = 2$), Neon ($Z = 10$), Argon ($Z = 18$), and Xenon ($Z = 54$). However, these materials containing noble gases are omitted since they are not relevant to the search of superhard materials. A plot of the number of materials of DB1M that contain a certain chemical element is shown in Figure 5.3.

Comparing Figures 5.1 and 5.3 it is possible to see that DB1M is less biased in terms of chemical elements than the database used to in the training process of the models. The fact that DB1M has materials with chemical elements with less representation in the train databases is not a problem. This only means that the model could have some problems predicting the properties of the materials containing these elements. The ability of the model to accurately predict the elastic properties of these materials will depend on the robustness of the model.

The best models trained with the databases presented in Section 5.1 and the predicted elastic properties for the materials in DB1M are presented and discussed in the next Chapter.

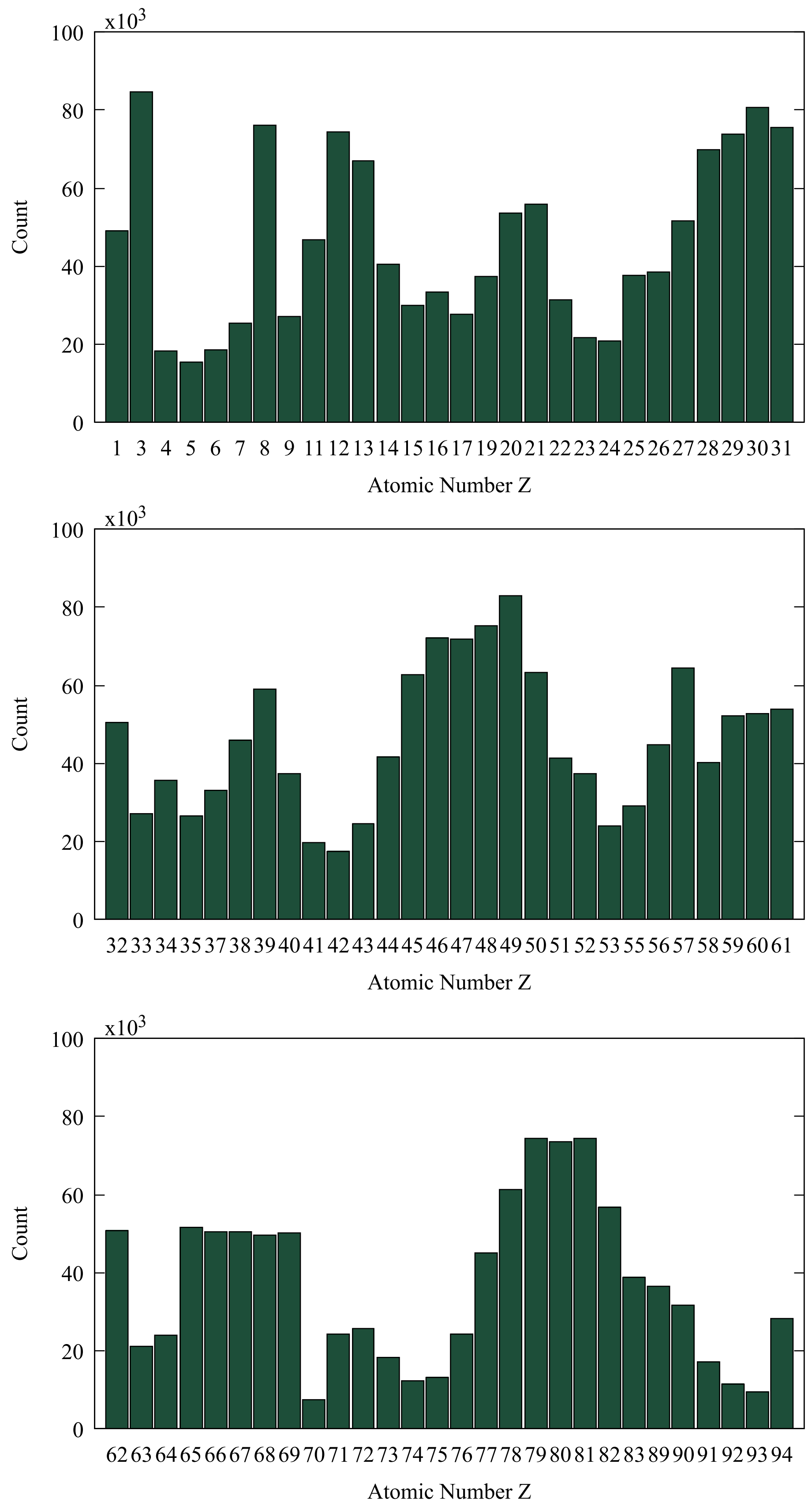


Figure 5.3: Representation of the number of materials containing a certain chemical element on the DB1M.

Chapter 6

Results and analysis

6.1 The training process

The models were trained using the database presented in Section 5.1. To choose the best model, a cross-validation scheme was applied to optimize the prediction of the elastic properties. Each model was trained with 60% of the data and then validated with 20% of the data, and the best-performing model in the validation set is selected [2, 3]. The remaining 20% of the data were used to test the predictive ability of the model. The cross-validation scheme was improved by merging it with the gridsearch method described in Section 4.3.

The train of each model was allowed to run through 400 epochs, i.e, the full train and validation sets were allowed to pass through the network 400 times. However, since the *Pytorch Lightning's* functionality `EarlyStopping` [30] was implemented, the process was stopped before the 400 epochs were completed, if a stopping criteria was met. The stopping criteria used was: if the loss function value $\mathcal{L}$ is the lowest and after 20 epochs remains in the interval $\left[\mathcal{L}_{\text{min}} - 0.001, \mathcal{L}_{\text{min}} + 0.001\right]$, then the training process stops.

The initial learning rate $\alpha = 0.02$ was decreased by a factor of 10 every 100 epochs, using the *Pytorch*'s routine `MultiStepLR` [54]. By not reducing the learning rate α at a continuous rate δ, this is $\alpha_{\text{new}} = \alpha - \delta$, it was possible to cover a larger space of learning rate's magnitudes, while saving computational resources.

Since the difference in magnitude between the largest and smaller value of the properties in study is huge (see Table 5.1), the models were trained using the decimal logarithm of the elastic properties. The use of the decimal logarithm works as a "normalization" of the values and helps to compare them. In fact, if the inputs of the network are not comparable the loss function will fluctuate heavily and affect the accuracy of the model [13]. Until there is an explicit indication, the values of the elastic properties presented were calculated using $\log\left(\frac{P}{P_0}\right)$, with P being the value of the elastic property in GPa and $P_0 = 1$ GPa.

The training and gridsearch process of the models was performed on a Intel i7-9700 with a Nvidia Tesla P100 (16 GB vRAM). For reference, the training and gridsearch process of the best model for the bulk modulus using the $\mathbf{v}_i^{(0)_{\text{orig}}}$, presented in the next section, took approximately 10 minutes.

6.2 Elastic properties - best model

The performance of the newly trained models was assessed by measuring the accuracy of the model's predictions in the test dataset. The accuracy was measured by making use of the mean absolute error (MAE) of the predicted values compared with DFT calculated values from the training database (see Section 5.1) and the R^2 metric. The predicted values are identified with the subscript ML and the values calculated with DFT are identified with the subscript DFT. A summary of the MAE obtained for each property model and a summary of benchmark MAE values obtained from [7, 8] are presented in Tables 6.1 and 6.2, respectively.

It is important to mention that the benchmark MAE values in Table 6.2 serve only as an indicator for the accuracy of the models trained in this thesis. In addition to being calculated differently, the datasets used to perform the benchmarks have neither the same materials nor the same number of materials as the dataset used to train and evaluate the accuracy of the models in this thesis. Therefore, the comparisons done in this Section are to be taken carefully and used as a representative value. It is not expected that the values calculated under the same conditions as those presented in this thesis using CGCNN v2019 [2, 3, 8] will differ much from those presented in Table 6.2.

As two models were trained for each elastic property, it is useful to define an easy way to distinguish them. Let the models trained with the $\mathbf{v}_i^{(0)_{\text{orig}}}$ atom feature vector be called Mv-O$^{(i)}$, with $i = G,\ K,\ H_{\text{v}}$, and the ones trained with the $\mathbf{v}_i^{(0)_{\text{Z}}}$ atom feature vector be called Mv-Z$^{(i)}$. In both cases i and represents the property for which the model was trained. For example, the shear modulus model trained with the $\mathbf{v}_i^{(0)_{\text{orig}}}$ atom feature vector is called Mv-O$^{(G)}$.

For the shear modulus G, the MAE of Mv-O$^{(G)}$ was 0.091 and the one for Mv-Z$^{(G)}$ was 0.091, respectively (see Table 6.1). This means that the accuracy of both models is similar. When compared with the benchmark MAE values in Table 6.2, both

Table 6.1: Summary of the accuracy of the predicted elastic properties on the test dataset for each atom feature vector. The mean value $\bar{y}$ of each property is included to give a feel for the magnitude of the MAE obtained.

(a) Predictions made using the models that used $\mathbf{v}_i^{(0)\text{orig}}$ as the atom feature vector.

Property	$\bar{y}$	MAE_{ML}	R^2
G	1.579	0.091	0.861
K	1.900	0.068	0.891
H_{v}	0.711	0.079	0.880

(b) Predictions made using the models that used $\mathbf{v}_i^{(0)\text{z}}$ as the atom feature vector.

Property	$\bar{y}$	MAE_{ML}	R^2
G	1.579	0.091	0.867
K	1.900	0.072	0.877
H_{v}	0.711	0.083	0.877

atom feature vectors seem to produce accurate models to predict the shear modulus G. Although these models are significantly more accurate than the Dummy model (MAE= 0.288) [8], their accuracy is still far from the one of the ALIGNN model (MAE= 0.071) [23].

The R^2 metric value obtained for Mv-O$^{(G)}$ and Mv-Z$^{(G)}$ was, respectively, 0.861 and 0.867 (see Table 6.1). These values combined with the analysis of the plots shown in Figure 6.1 indicate a good linear relation between the values G_{DFT} and G_{ML} for both Mv-O$^{(G)}$ and Mv-Z$^{(G)}$. It should be noted that Mv-Z$^{(G)}$ seems to produce fewer outliers than Mv-O$^{(G)}$.

As presented in Table 6.1, for the the bulk modulus K, the MAE of Mv-O$^{(K)}$ was 0.068 and the one of Mv-Z$^{(K)}$ was 0.072. By comparing these MAE values with the ones in Table 6.2, it is possible to conclude that both models can produce an accurate prediction for the bulk modulus K and are more accurate than the less accurate model in Table 6.2, the Dummy (MAE= 0.283) [8]. Just like for the accuracy of Mv-Z$^{(G)}$ and Mv-O$^{(G)}$, the accuracy of the models trained for the bulk modulus is still far from the accuracy of the MODNet model (MAE= 0.051) [55]. In contrast to what happened for the shear models, the model trained with the $\mathbf{v}_i^{(0)\text{orig}}$ atom feature vector is more accurate than the one the $\mathbf{v}_i^{(0)\text{z}}$ atom feature vector, as indicated by the MAE values.

For this property, the R^2 metric value for Mv-O$^{(K)}$ was 0.891 and the one for Mv-Z$^{(K)}$ was 0.877 has presented in Table 6.1. Combining the plot of each model shown in Figure 6.2 with its R^2 metric value, it is possible to see that both models have a good linear relation between the values K_{DFT} and K_{ML}. As for the shear modulus, the model trained using $\mathbf{v}_i^{(0)\mathrm{z}}$ as the atom feature vector appears to produce less outliers than the model trained using $\mathbf{v}_i^{(0)_{\mathrm{orig}}}$ as the atom feature vector.

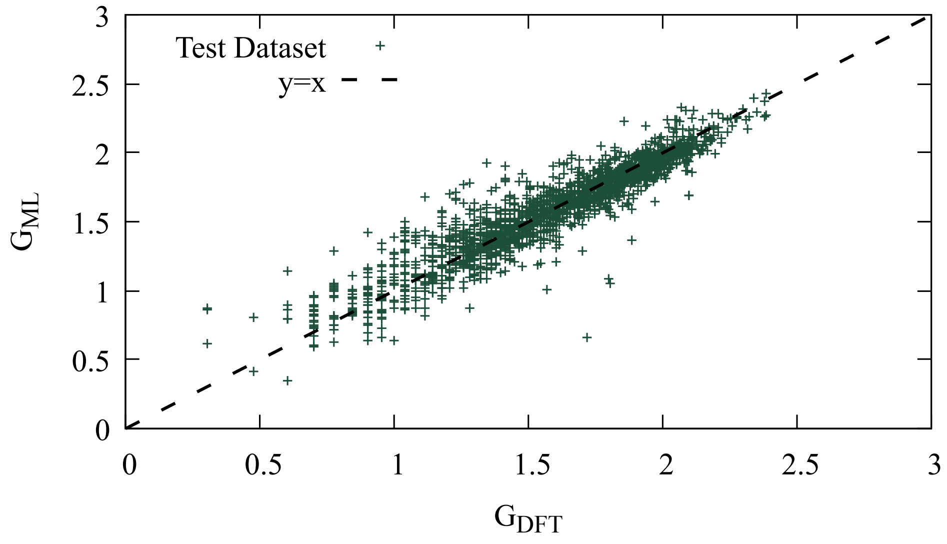
Finally, for Vickers' hardness H_{v}, the MAE of Mv-O$^{(H_{\mathrm{v}})}$ and Mv-Z$^{(H_{\mathrm{v}})}$ was, respectively, 0.079 and 0.083, respectively (see Table 6.1). As for the previous elastic property models, previous the model trained with the $\mathbf{v}_i^{(0)_{\mathrm{orig}}}$ atom feature vector is more accurate than the one trained with the $\mathbf{v}_i^{(0)\mathrm{z}}$ atom feature vector. For this property, the accuracy of both models is much lower than the accuracy of the Mazhnik and Oganov's model (MAE= 0.049) [7, 6]. It is worth noting that the MAE of Mazhnik and Oganov's model was calculated using the values in Table III of [7], therefore, the size of the dataset is different from the one used in this thesis.

For this property, the R^2 metric values obtained for Mv-O$^{(H_{\mathrm{v}})}$ and Mv-Z$^{(H_{\mathrm{v}})}$ were 0.880 and 0.877, respectively (see Table 6.1). As for the previous properties, the R^2 metric values combined with the analysis of the plots shown in Figure 6.3 indicate a good linear relation between the values $H_{\mathrm{v}_{\mathrm{DFT}}}$ and $H_{\mathrm{v}_{\mathrm{ML}}}$ for each model. From these plots it is also possible to see that Mv-Z$^{(H_{\mathrm{v}})}$ appears to produce less outliers than Mv-O$^{(H_{\mathrm{v}})}$.

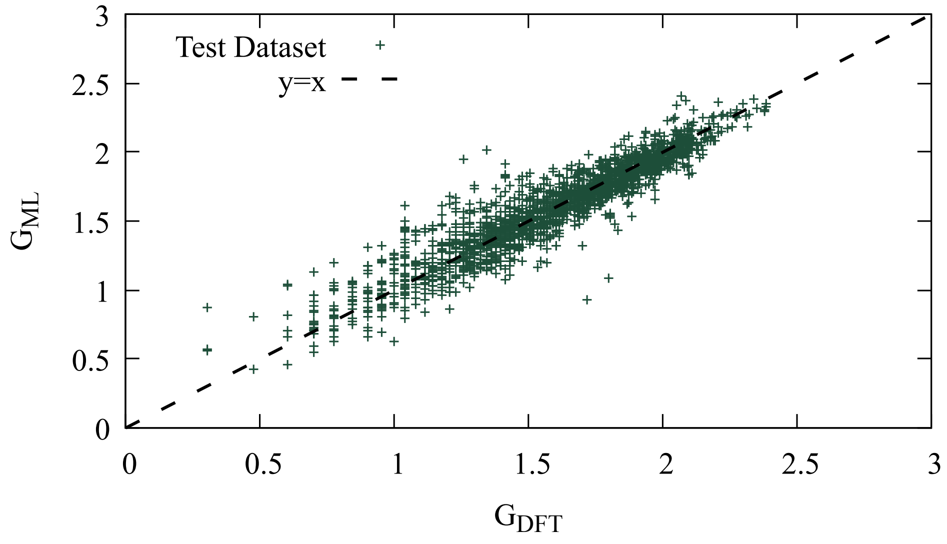
These results are a good indicator that, in future works, for large enough datasets, the model is able to build the atom feature vector using only the atomic number of the chemical element. This is important for elements to which there are not many experimental results available.

Table 6.2: Benchmark MAE values of other machine learning models used to predict elastic properties [7, 8].

Model	MAE$_G$	MAE$_K$	MAE$_{H_{\mathrm{v}}}$
CrabNet [19]	0.099	0.070	-
Finder v1.2 (composition only) [25]	0.098	0.074	-
Finder v1.2 (structure based) [25]	0.088	0.064	-
ALIGNN [23]	0.071	0.053	-
AMMExpress v2020 [8]	0.085	0.063	-
CGCNN v2019 [2, 3, 8]	0.087	0.067	-
Dummy [8]	0.288	0.283	-
MODNet [55]	0.072	0.051	-
RF-SCM/Magpie [8, 56–58]	0.102	0.078	-
Mazhnik and Oganov's model [6, 7]	-	-	0.049

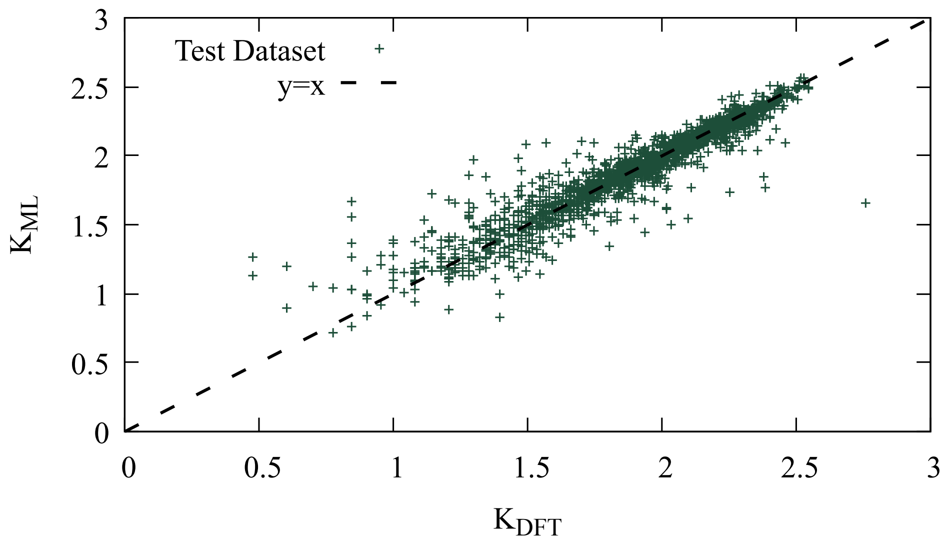


(a) Predictions made using the model Mv-O$^{(G)}$.

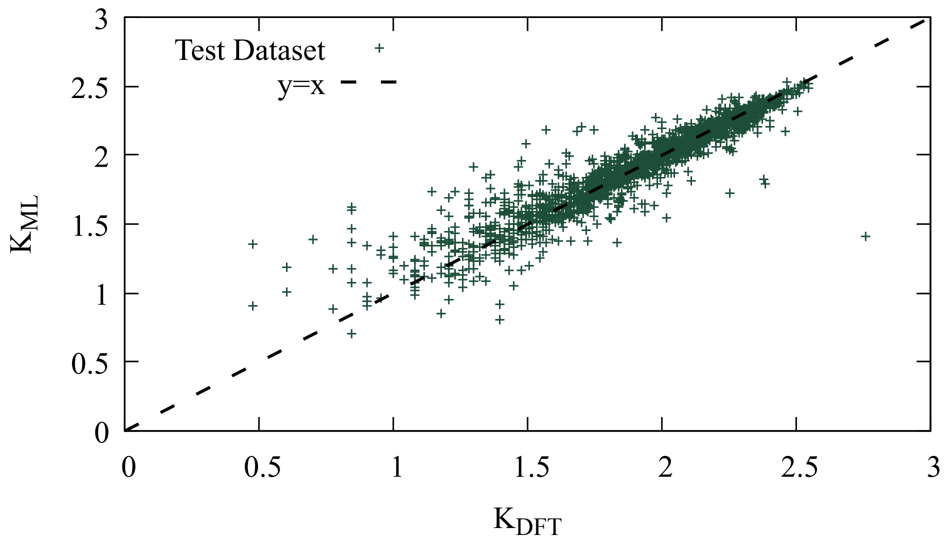


(b) Predictions made using the model Mv-Z$^{(G)}$.

Figure 6.1: Comparison of the predicted shear modulus G_{ML} and the DFT values G_{DFT} from [4] used as targets. The predictions were performed on the test dataset.

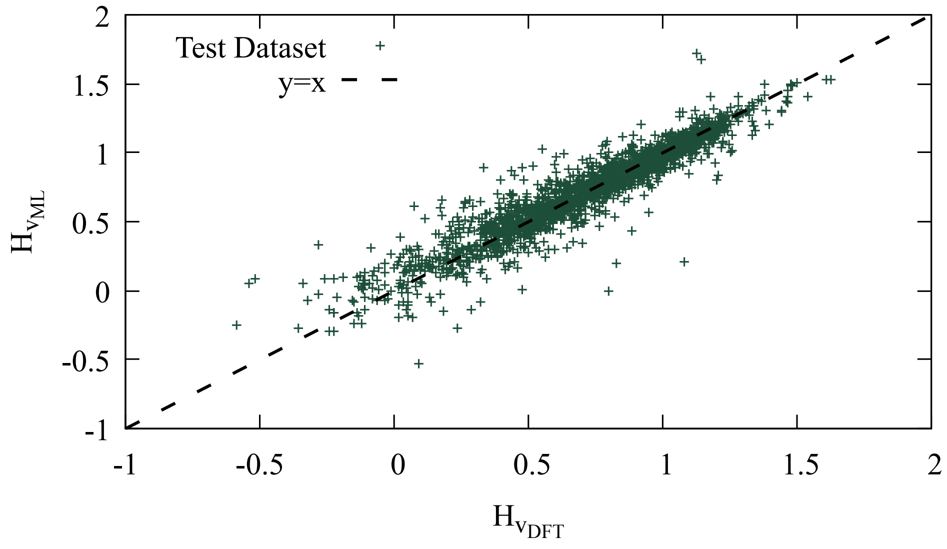


(a) Predictions made using the model Mv-O$^{(K)}$.

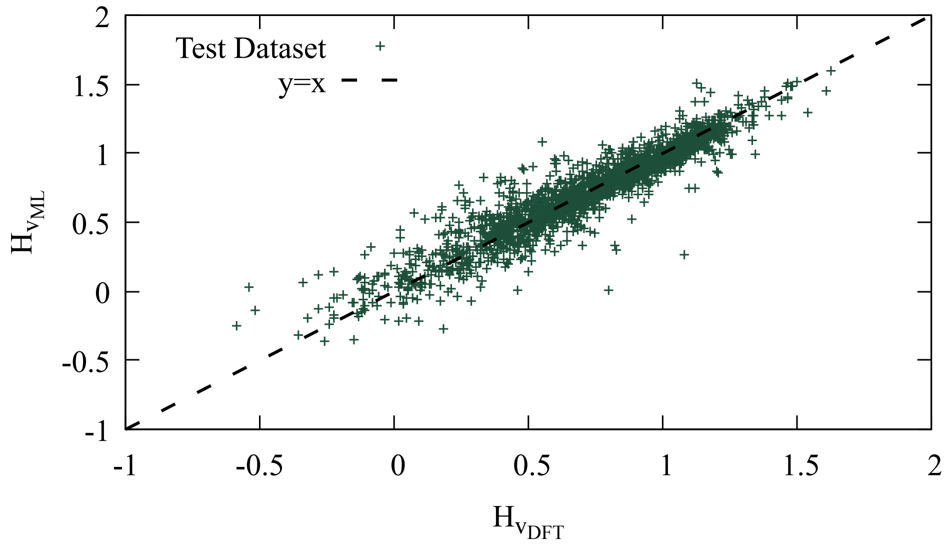


(b) Predictions made using the model Mv-Z$^{(K)}$.

Figure 6.2: Comparison of the predicted bulk modulus K_{ML} and the DFT values K_{DFT} from [4] used as targets. The predictions were performed on the test dataset.



(a) Predictions made using the model Mv-O$^{(H_v)}$.



(b) Predictions made using the model Mv-Z$^{(H_v)}$.

Figure 6.3: Comparison of the predicted Vickers' hardness $H_{v_{ML}}$ values and the values $H_{v_{DFT}}$ obtained as described in Section 5.2 used as targets.

6.2.1 The combined prediction

For the same property, the only difference between the models shown in the previous section is the atom feature vector. Then, it is possible to use an ensemble method to achieve better predictions. By using an ensemble method, it is possible to combine the predicted values of the two models trained for each property to have better predictions [59]. In this thesis, the ensemble method used was to consider the mean average between the values y^{orig} predicted by the model trained using the $\mathbf{v}_i^{(0)_{\mathrm{orig}}}$ atom feature vector and the values y^{Z} predicted by the model trained using the $\mathbf{v}_i^{(0)\mathrm{Z}}$ atom feature vector,

$$\bar{y} = \frac{y^{\mathrm{orig}} + y^{\mathrm{Z}}}{2}. \tag{6.1}$$

The combination of the predictions according with Eq. 6.1 will be treated as a model and called M2v-p throughout this thesis and will also be used in the analysis presented in Section 6.3.

The comparison between the MAE of the trained models and those used as a benchmark made in this Section must taken carefully and used as a representative value due to the same reasons discussed in the previous section.

For the shear modulus G, the MAE of M2v-p$^{(G)}$ was 0.087 (see Table 6.3). By using M2v-p$^{(G)}$, the accuracy of the predictions was improved to the point of becoming the fourth best model only behind the ALIGNN (MAE=0.071) [23], the AMMExpress v2020 (MAE=0.085) [8], and the Finder v1.2 (structure based) [25] (MAE=0.088) [25] (see Table 6.2). The M2v-p also produces a good linear relation between the values G_{DFT} and G_{ML} as can be seen in the plot shown in Figure 6.1 and by looking at the R^2 metric (R^2=0.875) presented in Table 6.3.

In the case of the bulk modulus K, the MAE of M2v-p$^{(K)}$ was 0.067 (see Table 6.3). Like for the shear modulus, the M2v-p$^{(K)}$ also improved the accuracy of the predictions to the point of becoming the fifth best model only behind the MODNet (MAE=0.051) [55], the ALIGNN (MAE=0.053), [23] the AMMExpress v2020 (MAE=0.063) [8], and the Finder v1.2 (structure based) (MAE=0.064) [25] (see Table 6.2). Similar to the previous property, the plot shown in Figure 6.4 for the bulk modulus and the R^2 metric value show a good linear relation between the values K_{DFT} and K_{ML}. The value of the R^2 metric obtained was 0.893 (see Table 6.3).

For Vickers' hardness H_{v}, the MAE of M2v-p$^{(H_{\mathrm{v}})}$ was 0.087 (see Table 6.3). This is a big improvement on the MAE of the trained models of the previous section. The accuracy remains lower than the accuracy of Mazhnik and Oganov's model [7, 6], but it is a step in the right direction. Finally, for this property, the plot shown in Figure 6.6 and the fact that the value of the R^2 metric obtain was 0.887 (see Table 6.3), indicate a good linear relation between the values $H_{\mathrm{v}_{\mathrm{DFT}}}$ and $H_{\mathrm{v}_{\mathrm{ML}}}$.

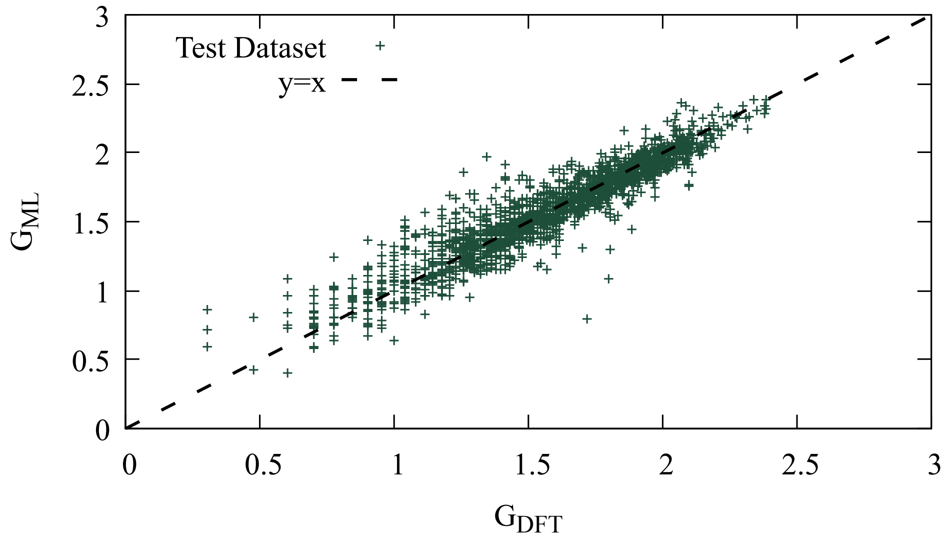


Figure 6.4: Comparison of the shear modulus G_{ML} values predicted with M2v-p$^{(G)}$ and the values G_{DFT} from [4] used as targets. The predictions were performed on the test dataset.

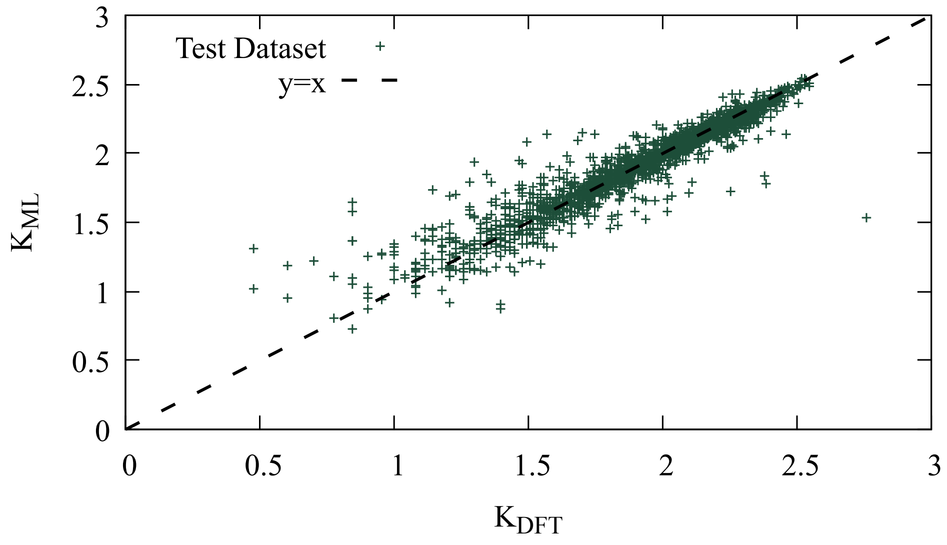


Figure 6.5: Comparison of the bulk modulus K_{ML} values predicted with M2v-p$^{(K)}$ and the values K_{DFT} from [4] used as targets. The predictions were performed on the test dataset.

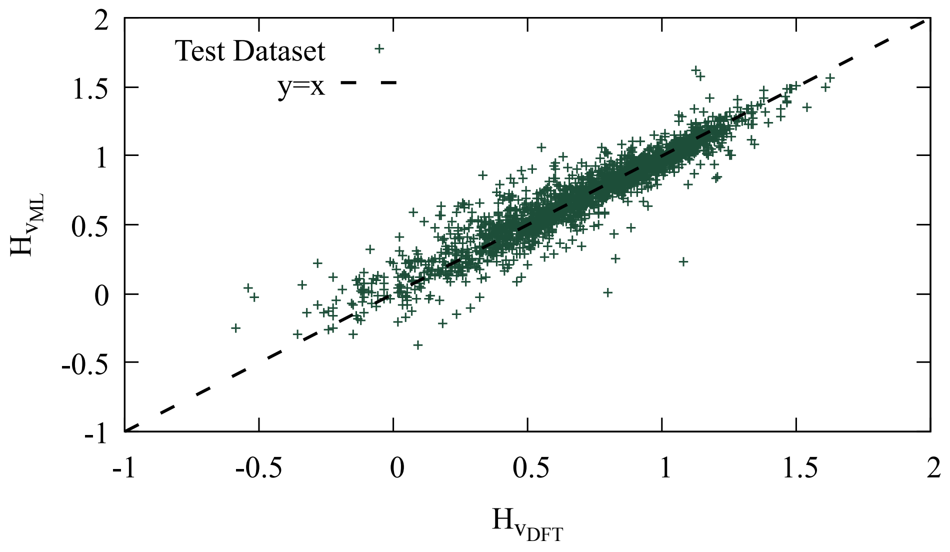


Figure 6.6: Comparison of the Vickers' hardness $H_{\mathrm{v_{ML}}}$ values predicted with M2v-p$^{(H_\mathrm{v})}$ and the values $H_{\mathrm{v_{DFT}}}$ obtained as described in Section 5.2 used as targets. The predictions were performed on the test dataset.

Table 6.3: Summary of the accuracy of M2v-p. The mean value $\bar{y}$ of each property is included to give a feel for the magnitude of the MAE.

Property	$\bar{y}$	MAE	R^2
G	1.579	0.087	0.875
K	1.900	0.066	0.893
H_v	0.711	0.077	0.887

6.3 Prediction of elastic properties of new materials

6.3.1 Distribution of the predicted values

The models presented in Section 6.2 were used to predict the elastic properties of the structures contained in the DB1M database introduced in Section 5.2. The final values of the predictions were obtained using the M2v-p model described in Section 6.2.1 and are presented below.

The distribution of the M2v-p$^{(G)}$'s predicted values for the shear modulus G is shown in Figure 6.7. These values are in the interval $[-0.267, 2.725]$ and have a mean and standard deviation value of 1.450 and 0.301, respectively.

The M2v-p$^{(K)}$'s predicted values distribution for the bulk modulus K is shown in Figure 6.8 and the values are in the interval $[0.473, 2.739]$. The mean and standard deviation value of the distribution are 1.797 and 0.319, respectively.

Finally, the distribution of the M2v-p$^{(H_\mathrm{v})}$'s predicted values for the Vickers' hardness H_v is shown in Figure 6.9. The predicted values have a mean and standard deviation value of 0.384 and 0.305, respectively, and are in the interval $[-0.985, 2.115]$.

The vertical lines in Figures 6.7 to 6.9 represent reference values of the elastic properties of Si, WC, c-BN and diamond. The value of the shear G_ML and bulk K_ML moduli are from Materials Project [9] and Vickers' hardness value $H_\mathrm{v}^\mathrm{exp}$ is a experimental value from [7] (Table 6.4).

Table 6.4: Summary of the elastic properties of some materials of interest [4, 7, 9]. The Materials Project's id of the materials is included to identify them.

Material	id	G_MP [4, 9]	K_MP [4, 9]	$H_\mathrm{v}^\mathrm{exp}$ [7]
Diamond	66	2.720	2.640	1.982
c-BN	1639	-	-	1.820
WC	1894	2.446	2.586	1.431
Si	149	1.785	1.919	1.079

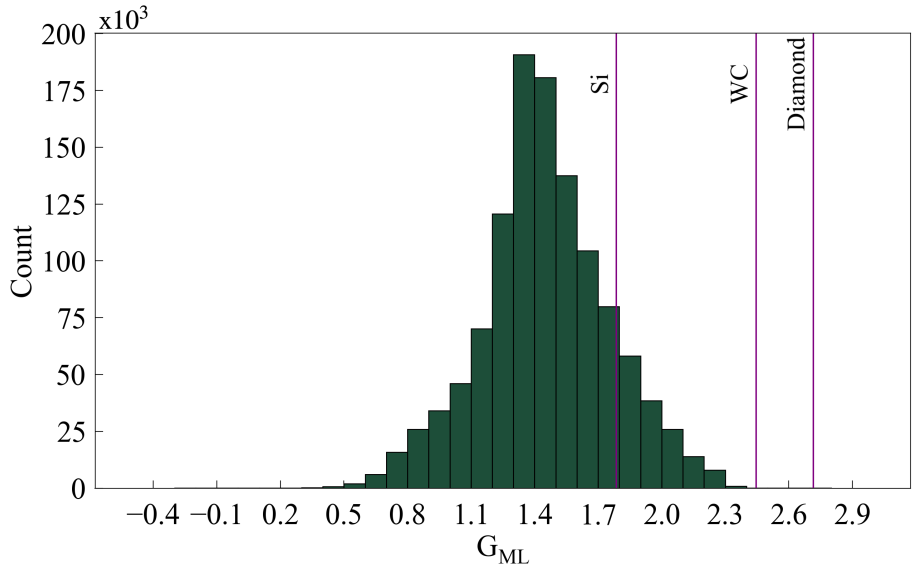


Figure 6.7: Distribution of the predicted shear modulus values G_ML of the materials contained in the DB1M database.

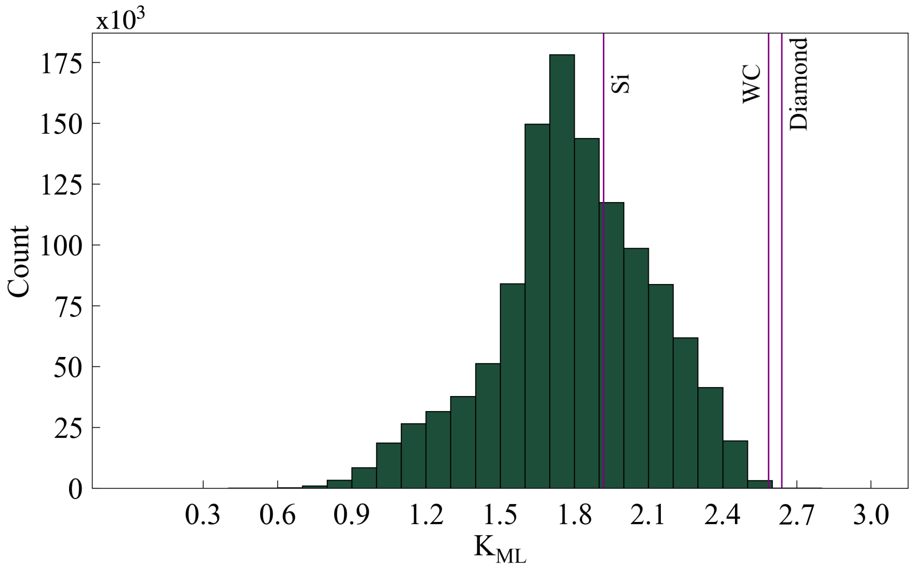


Figure 6.8: Distribution of the predicted bulk modulus values K_{ML} of the materials contained in the DB1M database.

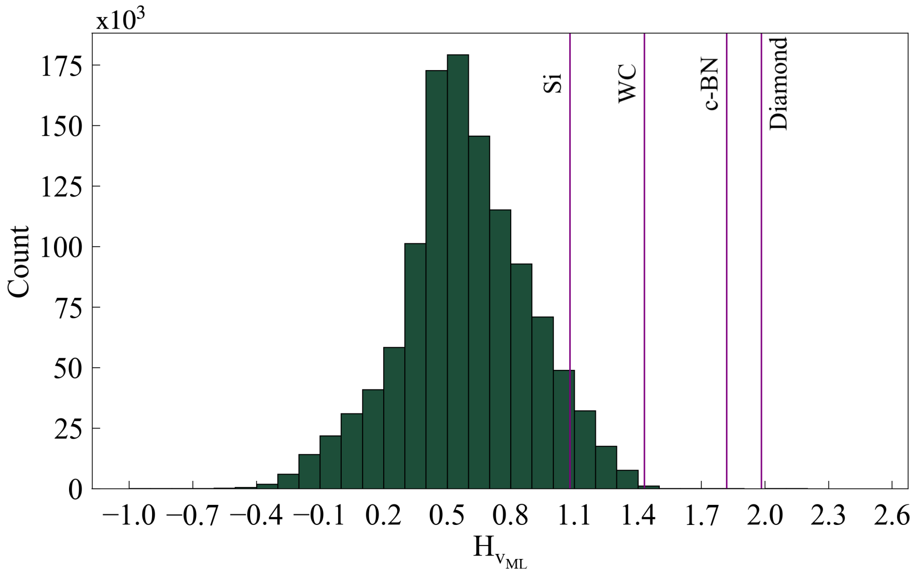


Figure 6.9: Distribution of the predicted Vickers' hardness values $H_{\mathrm{v_{ML}}}$ of the materials contained in the DB1M database.

6.3.2 Comparison between the predicted and reference values for Vickers' hardness

The goal of this thesis was to find super-hard materials in DB1M. Super-hard materials are materials with Vickers' hardness H_{v} above 40 GPa when measured by the Vickers hardness experimental test discussed in Section 2.5 [60].

Therefore, it is useful to know how many materials the M2v-p$^{(H_\mathrm{v})}$ predicted to be super-hard. To do so, the predicted Vickers' hardness values $H_{\mathrm{v}_\mathrm{ML}}$ were compared with some experimental reference values $H_\mathrm{v}^\mathrm{exp}$ from [7]. The M2v-p$^{(H_\mathrm{v})}$ predicted:

- 7 materials with a $H_{\mathrm{v}_\mathrm{ML}}$ greater than the experimental value $H_\mathrm{v}^\mathrm{exp}$ of diamond (mp-66) [7, 9];
- 14 materials with a $H_{\mathrm{v}_\mathrm{ML}}$ equal to the experimental value $H_\mathrm{v}^\mathrm{exp}$ of diamond (mp-66) [7, 9];
- 1 materials with a $H_{\mathrm{v}_\mathrm{ML}}$ greater than the experimental value $H_\mathrm{v}^\mathrm{exp}$ of c-BN (mp-1639) [7, 9], but smaller than the experimental value of diamond (mp-66) [7, 9];
- 685 materials with a $H_{\mathrm{v}_\mathrm{ML}}$ greater than the experimental value $H_\mathrm{v}^\mathrm{exp}$ of WC (mp-1894) [7, 9], but smaller than the experimental value of c-BN (mp-1639) [7, 9];

The summary of Vickers' hardness values of the reference materials used in the previous comparisons are presented in Table 6.5. Like in Table 6.4, the materials are identified using their Materials Project id [9]. From this section onwards, the values of the elastic properties will be presented in GPa for convenience reasons.

Table 6.5: Summary of the elastic properties of some materials of interest [4, 7, 9]. The Materials Project's id of the materials is included to identify them.

Material	id	$H_\mathrm{v}^\mathrm{exp}$ [GPa] [7]
Diamond	66	96
c-BN	1639	66
WC	1894	27

6.3.3 Validation via DFT calculations

Carrying out the validation via DFT calculations of the predictions made for 1 million materials is not viable. Therefore, for this thesis, the focus were materials with an higher chance of being synthesizable, i.e., materials with an energy above the convex hull E_hull bellow 50 meV[1]. From these, the 25 materials with highest predicted shear and bulk moduli were chosen to be analysed using DFT calculations. Only 29 DFT calculations were performed since most of the materials chosen due to the shear modulus predictions were the same as those chosen due to the bulk modulus predictions.

[1]A discussion about the thermodynamic stability of a material is presented in Appendix A.

The analysis of the 29 chosen materials consisted in calculating their stiffness tensor, also called elastic moduli tensor. These calculations were performed in VASP via `atomate`'s [61] workflows, using the corresponding default input parameters. Once the tensor was computed, all the elastic properties could be obtained [6, 62]. The list of materials used and their space group *spg*, number of atoms N, and predicted (ML) and calculated (DFT) shear G and bulk K moduli and Vickers' hardness H_{v} are presented in Table 6.6. Information about the thermodynamic stability of materials is also given in Table 6.6 through the energy above the convex hull E_{hull}. This way of evaluate the thermodynamic stability of materials is discussed in Appendix A.

Looking at the Table 6.6 it is possible to see that most of the materials chosen are metal borides, in particular with a generic formula of the type MNB_3. The only materials that were not present in both lists of selected materials where $BeNb_2B_3$, $BeNbB_2$, Ir_3Os_5, Nb_2ReB_2, Os_2RuW, $TaBeB_2$, $TaTiB_3$, and VReB.

The MAE was used to assess if the predictions were in good agreement with the results obtained through more accurate DFT calculations. The MAEs obtained, in GPa, for the shear and bulk moduli and the Vickers's hardness were 26, 9, and 5, respectively. However, presenting the MAEs by themselves does not provide full insight into whether the predictions are good or not. Therefore, the mean value $\bar{y}$ of all values calculated via DFT for each property were calculated. The values obtained were, in GPa, 197, 304, and 26 for the shear and bulk moduli and the Vickers' hardness, respectively. By taking into account the MAE and the mean value $\bar{y}$ of each property, it is possible to see that the best predictions where made for the bulk modulus. These MAE values and the mean values $\bar{y}$ for the three elastic properties are presented in Table 6.7.

This statement can also be confirmed using the R^2 metric. The values obtained for this metric were -0.166, 0.901, and -0.014 for the shear and bulk moduli and the Vickers' hardness, respectively. These values show that the bulk modulus is the only one that shows good linear relation between the values K_{DFT} and K_{ML}. These linear relation is also visible in Figure 6.11. The plots of the shear modulus and Vickers' hardness are shown in Figures 6.10 and 6.12, respectively. The R^2 metric values for these three elastic properties are presented in Table 6.7.

All 29 materials in Table 6.6 were considered thermodynamically stable or *quasi*-stable since all have energy above the convex hull E_{hull} bellow 50 meV and only the B_2OsW_2 was considered to be elastically unstable. A discussion about the elastic stability of a material is presented in Appendix A.

Table 6.6: Summary of the predicted (ML) and calculated (DFT) shear G and bulk K moduli and Vickers' hardness H_v for the 29 different materials chosen for validation via. Apart form the elastic properties, the table also includes information about the space group *spg*, number of atoms N and the energy above the convex hull E_{hull}. The materials are sorted in descending order of Vickers' hardness.

Material	*spg*	N	E_{hull} [meV]	G_{ML} [GPa]	G_{DFT} [GPa]	K_{ML} [GPa]	K_{DFT} [GPa]	$H_{v_{ML}}$ [GPa]	$H_{v_{DFT}}$ [GPa]
$TiVB_3$	63	10	16	226	245	270	262	36	42
$TaTiB_3$	63	10	14	209	230	262	268	33	36
Ta_2BeB_3	69	12	0	218	232	272	272	25	36
$BeNb_2B_3$	69	12	0	223	222	268	255	26	35
TiB_3W	63	10	0	216	235	297	293	33	33
$NbVB_3$	63	10	13	226	217	285	279	29	29
$TaVB_3$	63	10	13	224	222	289	290	29	29
Os_5Ru_3	25	8	18	215	236	356	365	27	27
$TaNbB_3$	63	10	0	230	213	285	288	28	27
Os_2Ru	15	6	14	213	233	360	369	28	27
$TaBeB_2$	62	16	39	209	192	248	248	25	26
$BeNbB_2$	62	16	29	208	186	244	235	25	26
$MoOs_4Ru$	13	12	21	210	220	358	357	28	25
Ir_3Os_5	25	8	28	203	196	365	378	28	25
V_2ReB_2	127	20	8	217	206	301	299	31	24
VOs_3	44	4	0	224	197	362	342	27	23
Os_2RuW	51	8	49	201	189	341	350	26	23
$MoOs_2$	62	12	38	209	188	358	345	28	23
V_2TcB_2	127	20	8	209	197	286	283	28	23
$VReB_2$	38	4	26	230	200	306	319	33	23
VB_2Mo	26	8	0	208	196	287	293	26	22
$TaTcB_3$	63	10	19	215	191	293	311	28	22
$VReB$	63	6	21	203	185	299	310	27	22
Nb_2ReB_2	127	20	0	206	189	295	290	24	22
$NbTcB_3$	63	10	4	222	174	293	294	27	20
$VTcB_2$	26	8	8	210	147	294	299	28	19
BIr	129	4	28	216	139	299	265	31	18
$VCoOs_2$	123	4	44	209	119	339	312	25	17
B_2OsW_2	128	20	46	208	106	364	352	26	16

Since the objective was to search for super-hard materials, it is worth mentioning the prediction of the super-hard ternary compound $TiVB_3$ with a Vickers' hardness of 42 GPa (see Table 6.6). There were also predicted some osmium compounds with high bulk modulus.

Table 6.7: Summary of the accuracy of the prediction of the elastic properties of the materials analysed with DFT calculations. The mean value $\bar{y}$ of all values calculated via DFT for each property is included to give a feel for the magnitude of the MAE obtained.

Metric	G	K	H_v	E	ν
$\bar{y}$	197 GPa	304 GPa	26 GPa	419 GPa	0.23
MAE	26 GPa	9 GPa	5 GPa	55 GPa	0.03
R^2	-0.166	0.901	-0.014	-0.241	0.411

As discussed in Section 2.3.1, it is possible to use Eqs. 2.25 and 2.26 to obtain, respectively, Young's modulus E and Poisson's ratio ν using the shear and bulk moduli. Let Young's modulus and Poisson's ratio calculated using the shear and bulk moduli obtained from DFT calculations be called E_DFT and ν_DFT, respectively. These elastic properties calculated using the predicted values of the shear and bulk moduli were called E_ML and ν_ML, respectively. Young's modulus and Poisson's ratio calculated using Eqs. 2.25 and 2.26 and both type of values, i.e, using DFT and predicted values, are presented in Table 6.8.

The MAE was used to check if Young's modulus and Poisson's ratio calculated with predicted values were in good agreement with the ones calculated with values obtain through DFT calculations. The MAEs obtained, in GPa, were 55 and 0.03 for Young's modulus and Poisson's ratio, respectively. The mean value $\bar{y}$ of all values calculated using via DFT for each property were calculated and the values obtained were, in GPa, 419 and 0.23 for Young's modulus and Poisson's ratio, respectively. Looking at the R^2 metric, the values obtained were -0.241 and 0.411 for Young's modulus and Poisson's ratio, respectively. This means that, despite not having the best linear relation between the values calculated using the model's predictions and the ones calculated with DFT's values, the best predictions were made for Poisson's ratio. The plots of Young's modulus and Poisson's ratio are shown in Figures 6.13 and 6.14, respectively.

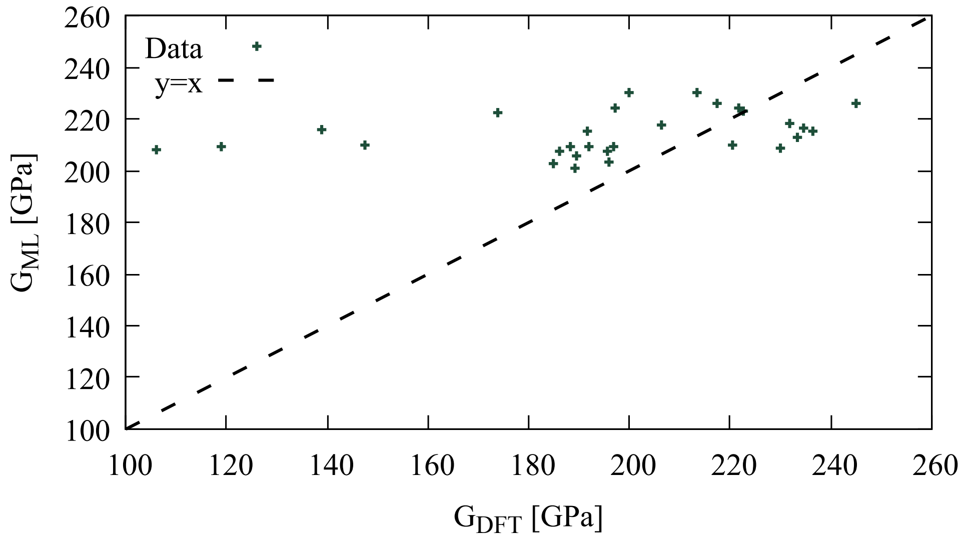


Figure 6.10: Comparison of the shear modulus G_{ML} values predicted with M2v-p$^{(G)}$ and the values G_{DFT} from DFT calculations used as targets for the 29 materials chosen to be analysed with DFT calculations.

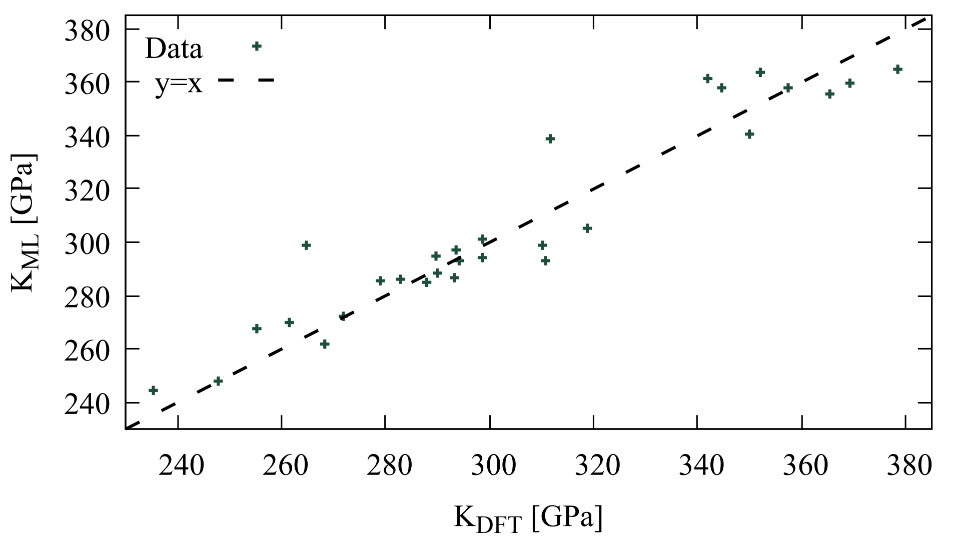


Figure 6.11: Comparison of the bulk modulus K_{ML} values predicted with M2v-p$^{(K)}$ and the values K_{DFT} from DFT calculations used as targets for the 29 materials chosen to be analysed with DFT calculations.

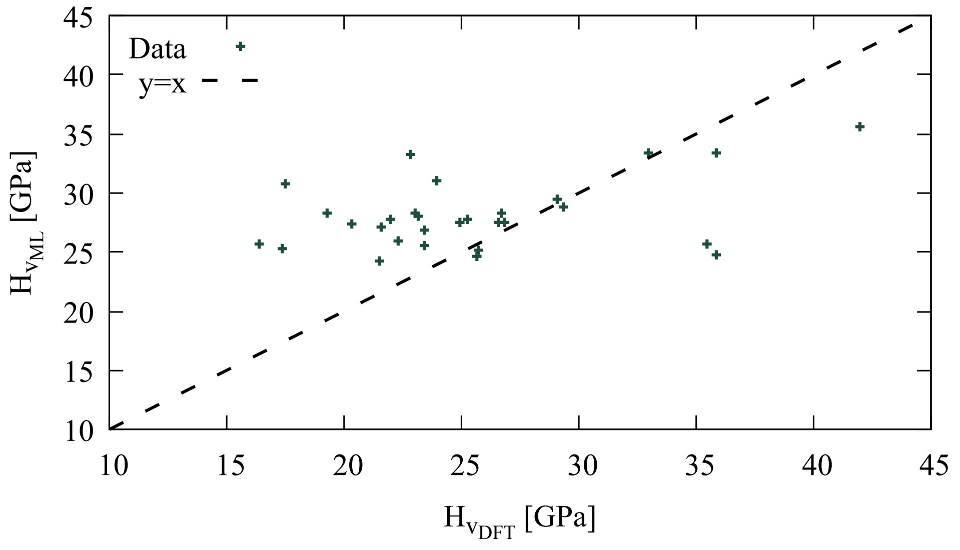


Figure 6.12: Comparison of Vickers' hardness $H_{\mathrm{v}_{\mathrm{ML}}}$ values predicted with M2v-p$^{(H_\mathrm{v})}$ and the values $H_{\mathrm{v}_{\mathrm{DFT}}}$ obtained as described in Section 5.2 used as targets for the 29 materials chosen for validation.

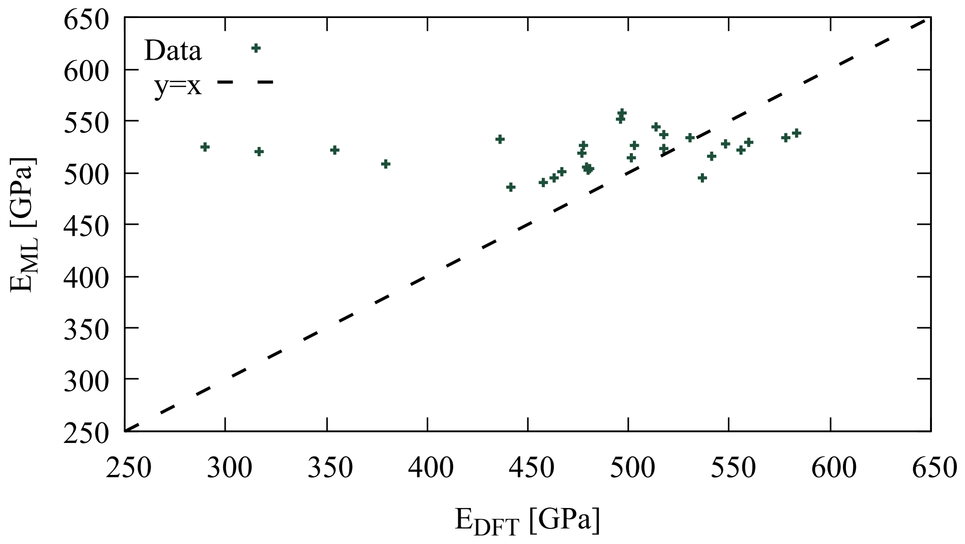


Figure 6.13: Comparison of Young's modulus E_{ML} values predicted with M2v-p and the values E_{DFT} obtained as described in Section 6.3.3 used as targets for the 29 materials chosen to be analysed with DFT calculations.

Table 6.8: Summary of Young's modulus E and Poisson's ratio ν calculated using predicted (ML) and calculated (DFT) shear G and bulk K moduli for the 29 different materials chosen for validation.

Material	E_{ML} [GPa]	E_{DFT} [GPa]	ν_{ML}	ν_{DFT}
$TiVB_3$	530	560	0.17	0.14
$TaTiB_3$	495	537	0.19	0.17
Ta_2BeB_3	516	541	0.18	0.17
$BeNb_2B_3$	523	517	0.17	0.16
TiB_3W	522	556	0.21	0.18
$NbVB_3$	537	518	0.19	0.19
$TaVB_3$	534	530	0.19	0.20
Os_5Ru_3	538	583	0.25	0.23
$TaNbB_3$	544	513	0.18	0.20
Os_2Ru	533	578	0.25	0.24
$TaBeB_2$	490	458	0.17	0.19
$BeNbB_2$	486	442	0.17	0.19
$MoOs_4Ru$	527	548	0.25	0.24
Ir_3Os_5	515	501	0.27	0.28
V_2ReB_2	526	503	0.21	0.22
VOs_3	557	496	0.24	0.26
Os_2RuW	504	481	0.25	0.27
$MoOs_2$	526	477	0.26	0.27
V_2TcB_2	505	479	0.21	0.22
$VReB_2$	552	496	0.20	0.24
VB_2Mo	502	480	0.21	0.23
$TaTcB_3$	518	477	0.21	0.24
VReB	496	463	0.22	0.25
Nb_2ReB_2	501	466	0.22	0.23
$NbTcB_3$	532	436	0.20	0.25
$VTcB_2$	509	380	0.21	0.29
BIr	522	354	0.21	0.28
$VCoOs_2$	520	317	0.24	0.33
B_2OsW_2	525	289	0.26	0.36

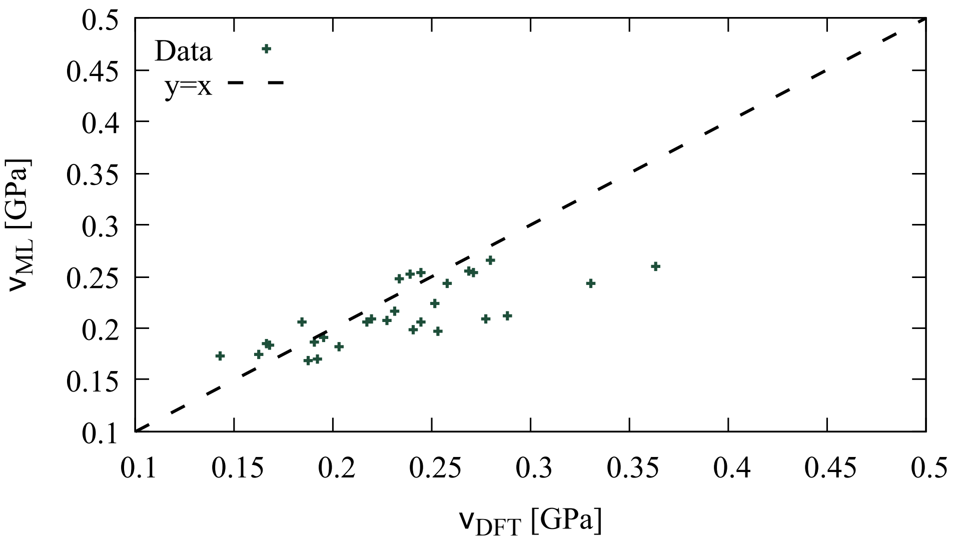


Figure 6.14: Comparison of Poisson's ratio ν_{ML} values predicted with M2v-p and the values ν_{DFT} obtained as described in Section 6.3.3 used as targets for the 29 materials chosen to be analysed with DFT calculations.

Chapter 7

Conclusions and future work

7.1 Conclusions

The goal of this thesis was to predict the shear and bulk moduli and Vickers' hardness with DFT accuracy of materials from a dataset of [1] in the search for super-hard materials.

Rewrite of the original code in *Pytorch Lightning* framework [30] and the implementation of the package *Optuna* [31] to fine-tune the model hyperparameters has proven to be useful in the management of the training of all models used.

From Sections 6.2, it is possible to see that the results obtained using the original atom feature vector and the one containing only the atomic number are similar. This is a good indicator that, in future works, for large enough datasets, the model is able to build the atom feature vector using only the atomic number of the chemical element. This is important for elements to which there are not many experimental results available.

The model trained for each property was able to produce accurate predictions on the data from DB1M. The search for super-hard materials in this dataset resulted in the prediction of the super-hard ternary compound $TiVB_3$ with a Vickers' hardness of 42 GPa. There were also predicted some osmium compounds with high bulk modulus.

7.2 Future work

In this thesis, the SGD optimiser was used to calculate the gradients in the training process. It would be important to test if the use of the ADAM optimiser, the standard optimizer, to calculate these gradients would affect the accuracy of the model.

In terms of the feature vectors used, it would be interesting to continue to investigate the importance of the features used to build the atom feature vector due to the results obtained. It would also be interesting to see the response of the model to the change in the bond feature vector.

Since the ability to remove the values of the network before the last activation function was implemented, it would be interesting to use these values as input features in simpler models to explore the accuracy of transferring the feature engineered using a huge dataset to small and specific datasets. This process was not tested in this thesis due to lack of time.

The model is now able to perform multi-class classification. In future work, the model could be trained and used to classify the materials of DB1M. It was not possible to perform this search due to lack of time.

The model shows good predictive capabilities for the elastic properties studied. It would be interesting to analyse the materials predicted to have high Vickers' hardness but with energy above the convex hull between 50 meV and 200 meV. In the future, the model should be applied to predict other properties of the materials from DB1M.

References

[1] Jonathan Schmidt, Noah Hoffmann, Pedro Borlido, Pedro J.M.A. Carriço, Tiago F. T. Cerqueira, Silvana Botti, and Miguel A. L. Marques. Large-scale machine-learning-assisted exploration of the material space. submitted, 2022.

[2] Tian Xie and Jeffrey C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 120:145301, Apr 2018.

[3] Tian Xie. *Deep learning methods for the design and understanding of solid materials*. PhD thesis, Massachusetts Institute of Technology, 2020.

[4] Maarten de Jong, Wei Chen, Thomas Angsten, Anubhav Jain, Randy Notestine, Anthony Gamst, Marcel Sluiter, Chaitanya Krishna Ande, Sybrand van der Zwaag, Jose J Plata, Cormac Toher, Stefano Curtarolo, Gerbrand Ceder, Kristin A. Persson, and Mark Asta. Charting the complete elastic properties of inorganic crystalline compounds. *Scientific Data*, 2(1):150009, 2015.

[5] L. D. Landau and E. M. Lifshitz. *Theory of Elasticity*, volume 7. Elsevier, 1986.

[6] Efim Mazhnik and Artem R. Oganov. A model of hardness and fracture toughness of solids. *Journal of Applied Physics*, 126(12):125109, 2019.

[7] Efim Mazhnik and Artem R. Oganov. Application of machine learning methods for predicting new superhard materials. *Journal of Applied Physics*, 128(7):075102, 2020.

[8] Alexander Dunn, Qi Wang, Alex Ganose, Daniel Dopp, and Anubhav Jain. Benchmarking materials property prediction methods: the matbench test set and automatminer reference algorithm. *npj Computational Materials*, 6(1), 2022.

[9] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[10] Pramila Shinde and Seema Shah. A review of machine learning and deep learning applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, 2018.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[12] Kamal Choudhary, Brian DeCost, Chi Chen, Anubhav Jain, Francesca Tavazza, Ryan Cohn, Cheol Woo Park, Alok Choudhary, Ankit Agrawal, Simon J. L. Billinge, Elizabeth Holm, Shyue Ping Ong, and Chris Wolverton. Recent advances and applications of deep learning methods in materials science. *npj Computational Materials*, 8(1):59, 2022.

[13] Charu C. Aggarwal. *Neural Networks and Deep Learning.* Springer, Cham, 2018.

[14] Scott Kirklin, James E Saal, Bryce Meredig, Alex Thompson, Jeff W Doak, Muratahan Aykol, Stephan Rühl, and Chris Wolverton. The open quantum materials database (oqmd): assessing the accuracy of dft formation energies. *npj Computational Materials*, 1(1):15010, 2015.

[15] Stefano Curtarolo, Wahyu Setyawan, Gus L. W. Hart, Michal Jahnatek, Roman V. Chepulskii, Richard H. Taylor, Shidong Wang, Junkai Xue, Kesong Yang, Ohad Levy, Michael J. Mehl, Harold T. Stokes, Denis O. Demchenko, and Dane Morgan. Aflow: An automatic framework for high-throughput materials discovery. *Computational Materials Science*, 58:218–226, 2012.

[16] Kamal Choudhary, Kevin Garrity, Andrew Reid, Brian DeCost, Adam Biacchi, Angela Hight Walker, Zachary Trautt, Jason Hattrick-Simpers, A. Kusne, Andrea Centrone, Albert Davydov, Jie Jiang, Ruth Pachter, Gowoon Cheon, Evan Reed, Ankit Agrawal, Xiaofeng Qian, Vinit Sharma, Houlong Zhuang, and Francesca Tavazza. The joint automated repository for various integrated simulations (jarvis) for data-driven materials design. 6:173, 12 2020.

[17] Valentin Stanev, Corey Oses, A. Gilad Kusne, Efrain Rodriguez, Johnpierre Paglione, Stefano Curtarolo, and Ichiro Takeuchi. Machine learning modeling of superconducting critical temperature. *npj Computational Materials*, 4(1):29, 2018.

[18] Dipendra Jha, Logan Ward, Arindam Paul, Wei-keng Liao, Alok Choudhary, Chris Wolverton, and Ankit Agrawal. Elemnet: Deep learning the chemistry of materials from only elemental composition. *Scientific Reports*, 8(1):17593, 2018.

[19] Anthony Yu-Tung Wang, Steven K. Kauwe, Ryan J. Murdock, and Taylor D. Sparks. Compositionally restricted attention-based network for materials property predictions. *npj Computational Materials*, 7(1):77, 2021.

[20] Quan Zhou, Peizhe Tang, Shenxiu Liu, Jinbo Pan, Qimin Yan, and Shou-Cheng Zhang. Learning atoms for materials discovery. *Proceedings of the National Academy of Sciences*, 115(28):E6411–E6417, 2018.

[21] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 05 2019.

[22] Cheol Woo Park and Chris Wolverton. Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery. *Phys. Rev. Materials*, 4:063801, Jun 2020.

[23] Kamal Choudhary and Brian DeCost. Atomistic line graph neural network for improved materials property predictions. *npj Computational Materials*, 7(1):185, 2021.

[24] Jonathan Schmidt, Love Pettersson, Claudio Verdozzi, Silvana Botti, and Miguel A. L. Marques. Crystal graph attention networks for the prediction of stable materials. *Science Advances*, 7(49):eabi7948, 2021.

[25] Achintha Ihalage and Yang Hao. Formula graph self-attention network for representation-domain independent materials discovery. *Advanced Science*, 9(18):2200164, 2022.

[26] Xipeng Wang, Simón Ramírez-Hinestrosa, Jure Dobnikar, and Daan Frenkel. The lennard-jones potential: when (not) to use it. *Phys. Chem. Chem. Phys.*, 22:10624–10633, 2020.

[27] Nadezhda Chistyakova and Thi My Hue Tran. A study of the applicability of different types of interatomic potentials to compute elastic properties of metals with molecular dynamics methods. *AIP Conference Proceedings*, 1772(1):060019, 2016.

[28] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

[29] Chi Chen and Shyue Ping Ong. A universal graph deep learning interatomic potential for the periodic table, 2022.

[30] William Falcon et al.. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 2019.

[31] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[32] A.B. Bhatia and R.N. Singh. *Mechanics of Deformable Media*. Graduate student series in physics. Hilger, 1986.

[33] M. T. Dove. *Structure and Dynamics An Atomic View of Materials*, volume 1. Oxford University Press, 2003.

[34] Xing-Qiu Chen, Haiyang Niu, Dianzhong Li, and Yiyi Li. Modeling hardness of polycrystalline materials and bulk metallic glasses. *Intermetallics*, 19(9):1275–1281, 2011.

[35] Faming Gao, Julong He, Erdong Wu, Shimin Liu, Dongli Yu, Dongchun Li, Siyuan Zhang, and Yongjun Tian. Hardness of covalent crystals. *Phys. Rev. Lett.*, 91:015502, Jul 2003.

[36] Antonín Šimůnek and Ji ří Vackář. Hardness of covalent and ionic crystals: First-principle calculations. *Phys. Rev. Lett.*, 96:085501, Mar 2006.

[37] Keyan Li, Xingtao Wang, Fangfang Zhang, and Dongfeng Xue. Electronegativity identification of novel superhard materials. *Phys. Rev. Lett.*, 100:235504, Jun 2008.

[38] Andriy O. Lyakhov and Artem R. Oganov. Evolutionary search for superhard materials: Methodology and applications to forms of carbon and tio$_2$. *Phys. Rev. B*, 84:092103, Sep 2011.

[39] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning.* arXiv, 2021.

[40] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.

[41] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[42] Sebastian Bock, Josef Goppold, and Martin Georg Weiß. An improvement of the convergence proof of the adam-optimizer. *CoRR*, abs/1804.10587, 2018.

[43] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[44] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[45] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.

[46] Hamed Habibi Aghdam and Elnaz Jahani Heravi. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification.* Springer International Publishing, 1 edition, 2017.

[47] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629, 2018.

[48] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *CoRR*, abs/1502.02127, 2015.

[49] Mohammad Noor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Systematic ensemble model selection approach for educational data mining. *Knowledge-Based Systems*, 200:105992, 2020.

[50] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[51] Joohwi Lee and Ryoji Asahi. Transfer learning for materials informatics using crystal graph convolutional neural network. *Computational Materials Science*, 190:110314, 2021.

[52] Soumya Sanyal, Janakiraman Balachandran, Naganand Yadati, Abhishek Kumar, Padmini Rajagopalan, Sanyal Suchismita, and Partha Talukdar. Mt-cgcnn: Integrating crystal graph convolutional neural network with multitask learning for material property prediction, 11 2018.

[53] Luis M. Antunes, Ricardo Grau-Crespo, and Keith T. Butler. Distributed representations of atoms and materials for machine learning. *npj Computational Materials*, 8(1):44, 2022.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[55] Pierre-Paul De Breuck, Matthew L Evans, and Gian-Marco Rignanese. Robust model benchmarking and bias-imbalance in data-driven materials science: a case study on MODNet. *Journal of Physics: Condensed Matter*, 33(40):404002, jul 2021.

[56] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[57] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):16028, 2016.

[58] Felix Faber, Alexander Lindmaa, O. Anatole von Lilienfeld, and Rickard Armiento. Crystal structure representations for machine learning models of formation energies, 2015.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[60] Georgiy Akopov, Lisa E. Pangilinan, Reza Mohammadi, and Richard B. Kaner. Perspective: Superhard metal borides: A look forward. *APL Materials*, 6(7):070901, 2018.

[61] Kiran Mathew, Joseph H. Montoya, Alireza Faghaninia, Shyam Dwarakanath, Muratahan Aykol, Hanmei Tang, Iek heng Chu, Tess Smidt, Brandon Bocklund, Matthew Horton, John Dagdelen, Brandon Wood, Zi-Kui Liu, Jeffrey Neaton, Shyue Ping Ong, Kristin Persson, and Anubhav Jain. Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows. *Computational Materials Science*, 139:140–152, 2017.

[62] R. Hill. The elastic behavior of a crystalline aggregate. *Proceedings of the Physical Society. Section A*, 65:349, 12 2002.

[63] T.F.T. Cerqueira. *“Structural prediction and materials design: from high throughput to global minima optimization methods,”*. PhD thesis, Jena, 2017.

[64] Max Born. On the stability of crystal lattices. i. *Mathematical Proceedings of the Cambridge Philosophical Society*, 36(2):160–172, 1940.

[65] Félix Mouhat and François-Xavier Coudert. Necessary and sufficient elastic stability conditions in various crystal systems. *Phys. Rev. B*, 90:224104, Dec 2014.

Appendix A

Evaluate the stability of a material

A.1 Structural stability

A way to describe the structural stability of a material is to use the energy above the convex hull E_{hull}. The concept of convex hull arises from the need to guarantee that the decomposition channel of a material is the more stable channel available [63]. In other words, it is necessary to know if, for example, the material A_xB_x is not more stable than the material A_xB_y.

The convex hull of thermodynamic stability is the hypersurface in composition space that passes by all materials that are thermodynamically stable [63]. Therefore, a material is said to be:

- thermodynamically stable if its distance to the convex hull is zero or sufficiently close to zero;
- thermodynamically unstable if its distance to the convex hull above zero.

A.2 Elastic stability

As discussed in Chapter 2, the behaviour of a crystal is described by the elastic moduli tensor λ_{ijkl}. Then, as noted by Born [64], the properties of this tensor can be used to to determine the elastic stability of a crystal. In [65] are stated the following necessary and sufficient stability conditions:

- The tensor λ_{ijkl} is definite positive;
- all eigenvalues of λ_{ijkl} are positive;

which are possible formulations of the generic Born elastic stability conditions for an unstressed crystal. These conditions are agnostic to the symmetry of the crystal.