UCLouvain

École polytechnique de Louvain

Vibrational Properties of Solids

A Machine Learning Approach

Author: **Pierre-Paul De Breuck**
Supervisor: **Gian-Marco Rignanese**
Readers: **Christophe De Vleeschouwer, Geoffroy Hautier**
Academic year 2018–2019
Master [120] in Physical Engineering

Abstract

This work presents a novel approach based on a neural network and transfer learning for the prediction of thermodynamic properties of crystalline solids. It has the unique property of taking advantage of information stored across different properties, in order to increase overall accuracy. Indeed, vibrational properties of solids are key for material design and discovery. For instance, the vibrational entropy can easily alter the structure stability. Although ab initio high-throughput calculations allow for the screening of properties of thousands of materials, fast characterization and prediction of thermodynamic properties of materials is still missing due to the inherent computational cost involved in obtaining second-order derivatives in energy. Thanks to the presented thermal transfer neural network, a fast on-the-fly prediction of the vibrational entropy, energy, specific heat and Helmoltz free energy at various temperatures are possible and this for any crystal structure with a mean absolute error three times as low than previous models. Furthermore, this dissertation proposes an interpretable feature selection algorithm based on a dynamic relevance-redundancy criterion. Finally, trends are found between common material features and important causes to the vibrational entropy are stated, such as the radial environment of a structure's sites.

Acknowledgements

First and foremost I would want to sincerely thank my promotor, Prof. Gian-Marco Rignanese, for his help and advise throughout this work. A work that principally started as a personal project that he welcomed with enthusiasm, and I am really grateful for this.

I also want to thank Prof. Christophe De Vleeschouwer, who moreover inspired me with many ideas in the image recognition field, and Prof. Geoffroy Hautier in advance for reading this dissertation.

To my family and friends: many thanks for the support during the process!

Contents

Introduction

Materials are at the heart of our society. Be it structural materials such as steel or concrete that bear much of human constructions, be it silicon that has revolutionized the modern technologies or be it carbon nanotubes that offers unique attributes to the biomedical research, materials have a big impact on our society. From the Stone Age, through the Bronze and Iron Ages, to the modern silicon ara, it has always been of high importance to develop novel materials that overcome the current challenges faced by the civilization. Furthermore, it has been estimated that materials development enabled two-thirds of all advancements in computation over the past 40 years, and transformed other industries as well, such as energy storage [1].

Novel materials design and development are often a tedious process, and bringing them to the market takes approximately 20 years. Moreover, once a material has been adopted, it is rarely replaced within a short time span owing to the high cost associated to production infrastructure. Therefore introducing high-performance materials is a key factor for the success of many technological niches in demand of potential materials.

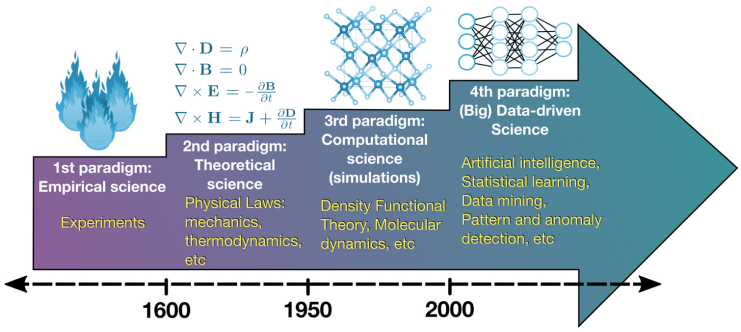

Figure 0.0.1: The 4 paradigms in science. Empirical, theoretical, computational, and data-driven. Each paradigm both benefits from and contributes to the others. Image taken from Ref. [2].

In order to address these problems, the materials science field emerged aiming at the characterization of materials through process, structure, and properties. Throughout the years, it has benefit and shaped the four science paradigms, see figure 0.0.1. Initially, most knowledge was gained through experiments which empirically showed the advantages of some materials to others. Later, the emergence of a theoretical basis enabled one to rationally design new compounds. In particular thanks to the formulation of Quantum Mechanics it was possible to understand the microscopic properties such as the concept of bond between atoms. However, a precise resolution of this latter for common systems is very difficult, as stated by Dirac in 1929 [3]:

'The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known, and the difficulty lies only in the fact that application of these laws leads to equations that are too complex to be solved'

Major efforts were then put on computational aspects rather than theoretical ones. Therefore, approximations to the Schrödinger equation were introduced, such as the Born-Oppenheimer approximation or the Hartree-Fock method. Together with the rise of computational machines, a quantum computational model arose known as the Density Functional Theory (DFT). It treats many-body systems by the use of functionals, i.e. functions of functions, by stating that the electronic density of any system determines all ground-state properties of the system (proposed and demonstrated by Hohenber and Kohn in 1964, see Ref. [4]). Since its initial developments, DFT has evolved to an accurate predictive algorithm, capable of predicting various properties of materials including drug design, solar cells and water splitting materials among others. It can therefore be seen as the third paradigm of science.

In parallel, during the 1950s, an important field within artificial intelligence (AI) emerged known as machine learning (ML), inspired by developments in statistics, computer science and technology, and neuroscience. It consists of many algorithms that are able to find patterns and relation in data, ranging from simple linear regressions to complex deep neural networks. By extracting knowledge from data they are able to predict unknown entries or to assist in decision-making processes under uncertainty. They learn in the sense that they will get better at a certain task by increasing the training set size without being explicitly programmed. Although the theory behind most of these models already existed for more than 50 years, they gained in popularity lately (and especially deep learning) partially thanks to the increase in computational power and partially thanks to the increase in the available data. A broader introduction to ML will be given in the first chapter.

The success of DFT together with high-throughput methods and the development of computational capabilities resulted in a huge amount of data consisting in many materials with their corresponding properties. This together with previously existing (and still increasing) experimental data naturally led to material databases such as the Materials Project [5]. The knowledge contained in these databases are really enormous, and could be beneficially used by ML-algorithms. This led to the emergence of the fourth paradigm of science: big data-driven science. It combines both fields of material science and machine learning.

A major advantage of ML-methods trained on material data is the fast on-the-fly prediction of material properties. Conventional first-principles approaches need to solve a series of self-consistent equations that are often very computational intensive and thus rather slow. Moreover, ML offers the ability to gain new insights into materials by discovering patterns and relations in the data.

So far, the field of molecular (primarily pharmaceutical and medicinal) science have experienced a stronger degree of uptake in ML compared to crystalline solids. This is partially due to the difficulty of representing crystals into a ML-suitable format and partially due to the high applicability of molecular science in the pharmaceutical industry [6], [7]. The application of machine learning for functional materials is an emerging field, and much has yet to be learned. In figure 0.0.2, the existing gap between stored data, information and captured knowledge is depicted. Clearly, using data-driven approaches is paramount in order to reduce this gap and advance the current research [2].

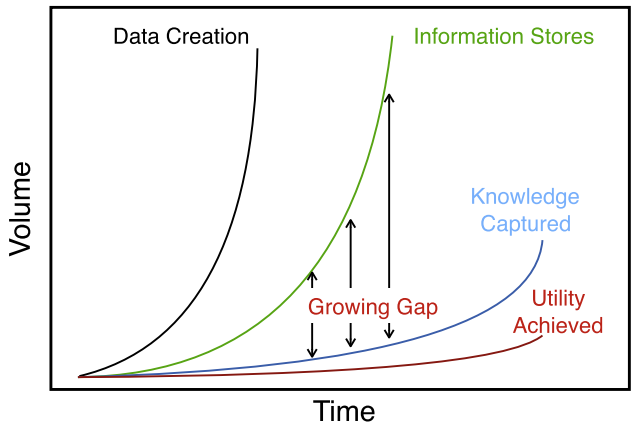

Figure 0.0.2: Illustration of the current growing gap between stored information and captured knowledge in the material science field. Image taken from Ref. [2].

In summary, given the importance of material characterization and the available knowledge in current databases that has not been captures yet, machine learning is key to offer a variety of new solutions to the materials science field. As a first step to this, the present work will introduce a novel framework to predict thermodynamic properties for crystalline solids.

Chapter 1 briefly motivates the importance of the thermodynamic properties and proposes a short reminder on machine learning. Chapter 2 contains a brief literature study on the subject. Chapter 3 discusses the process of featurizing a material, i.e. transforming it to a fixed-length vector. Chapter 4 analyzes the different features and targets based on their mutual relations. A new feature selection algorithm is proposed too based on the relevance-redundancy criterion, in order to find the best suited features for our task. Finally, Chapter 5 will propose a novel approach based on this selection algorithm and transfer learning in order to predict vibrational thermal data.

Chapter 1

Machine learning for material discovery

1.1 Introduction

The purpose of this introductory chapter is double. First, the goal of the present work, i.e. effective prediction of vibrational thermodynamic properties, is first motivated and detailed in the first section. Then, the second section will briefly refresh some fundamental Machine Learning concepts, crucial for the understanding of the remaining chapters.

1.2 Machine learning for vibrational properties of solids

As discussed in the introduction of this work, functional materials are found in numerous applications and are at the heart of many devices. The main challenge for the material scientist is to design novel compounds with desired properties, for a particular application. For example, concerning photovoltaics, one targets compounds with a specific bandgap. In order to do so, they can now rely on first-principles computations, which can guide the experimental identification and synthesis and hence reduce the development time and costs.

Over the last decade, first-principles calculations of materials properties, based on the density functional theory (DFT), have undergone both qualitative and quantitative changes of paradigm. At the qualitative level, modern programs, like ABINIT [8], and related atomic data provide nowadays access to intrinsic materials properties with a degree of precision comparable to experiment. At the quantitative level, high-throughput (HT) calculations based on these frameworks allow one to screen the properties of thousands of existing or hypothetical materials and store them in public materials databases which can then be intelligently interrogated (such as AFLOW [9], the Materials Cloud [10], the Materials Project [5], the Open Quantum Materials Database, etc.). HT calculations already made it possible to identify new compounds for a variety of applications (e.g. lithium battery and photovoltaic). However, investigations have so far been restricted to a limited set of properties (e.g. structure, internal energy, bandgaps, etc.). Clearly, the characterization and prediction of thermodynamic properties of materials is still missing today.

Important properties include the specific heat capacity at constant volume/pressure (C_V or C_p), the mode resolved and average Gruneisen parameters (γ_{qj} and γ) ,the thermal expansion coefficient (α_V), the Debye temperature (θ_D), the lattice thermal conductivity (κ_l), the vibrational entropy an the Gibbs free energy ($G(p,T)$ and $G(p,T,V)$). These properties are of great importance for the material scientist. For instance, the vibrational entropy may easily alter the ordering of the energies of the structures and modify which one is the most stable as the temperature is raised. Such properties can be calculated from first principles using density-functional perturbation theory (DFPT). The temperature dependence can first be accounted for within the harmonic approximation. Furthermore, within the quasiharmonic approximation, the implicit volume dependence through the frequencies is also taken into account : the thermodynamic potentials are thus calculated at a few volumes and an interpolation is performed in between. However, while ab initio relaxed crystal structures and groundstate energies are routinely provided in the HT databases, the thermal properties are much scarcer. This is because computing inter-atomic force constants (IFCs), i.e. the second derivatives of the energy with respect to the atomic displacements, is very demanding. Thus, in materials design, the effect of temperature is often accounted for by simple approximations.

Recently, Petretto *et al.* [11] extended the content of the Materials Project to phonon dispersion curves, including approximately 1500 entries at the DFT level relying on the generalized gradient approximation (GGA). The knowledge contained in this dataset could certainly be beneficially used by ML-algorithms in order to tackle the previously stated problem. So far, reported ML predictions on these properties have been obtained on very limited datasets and with minor effort on the ML aspect.

This work will therefore set as goal the prediction of thermodynamic data such as the vibrational entropy, Helmoltz free energy, specific heat and vibrational contribution to the internal energy. The idea is to have a fast and effective model that predicts thermodynamic properties at different temperatures for *any* crystal compound. This will make it possible to determine the effect of the vibrational entropy on the relative stability of different compounds, and this for many compounds. If seen to be working well, the proposed model could certainly be used on the remaining thermal properties.

The main dataset that will be used in this work, i.e. the high throughput study based on (DFPT) in the harmonic approximation (see Ref. [11]), contains four thermodynamic properties at various temperatures. These properties can all be derived from the phonon density of states (DOS) g(ω),

$$g(\omega) = \frac{1}{3nN} \sum_{\mathbf{q},l} \delta(\omega - \omega(\mathbf{q}, l)), \tag{1.1}$$

where n is the number of atoms per unit cell, N is the number of unit cells, q the wave vector and l the phonon mode.

From the DOS, several thermodynamic quantities can be obtained and are briefly defined hereunder [11].

Vibrationnal entropy

The vibrational entropy can be computed within the harmonic approximation as,

$$S = 3nNk_B \int_0^{\omega_L} \left(\frac{\hbar\omega}{2k_BT} \coth\left(\frac{\hbar\omega}{2k_BT}\right) - \ln\left(2\sinh\frac{\hbar\omega}{2k_BT}\right)\right) g(\omega)d\omega \tag{1.2}$$

where k_B is the Boltzmann constant and ω_L is the largest phonon frequency.

Physically, it represents the disorder (i.e. number of possible configurations over the lattice) of the phonons. Note that most solids also have an additional configurational entropy, due to the different species, which adds up to the total entropy of a compound. We only focus on the vibrational contribution in this work.

Internal energy

The vibrational internal energy can be computed within the harmonic approximation as,

$$\Delta U_{\text{ph}} = 3nN\frac{\hbar}{2} \int_0^{\omega_L} \omega \coth\left(\frac{\hbar\omega}{2k_BT}\right) g(\omega)d\omega \tag{1.3}$$

where k_B is the Boltzmann constant and ω_L is the largest phonon frequency. It respresents the energy carried by the phonons. Again, this latter only includes the vibrational part, and does not take into account the potential energy between atoms.

Helmoltz free energy

The Helmoltz free energy can be computed within the harmonic approximation as,

$$\Delta F = 3nNk_BT \int_0^{\omega_L} \ln\left(2\sinh\frac{\hbar\omega}{2k_BT}\right) g(\omega)d\omega. \tag{1.4}$$

where k_B is the Boltzmann constant and ω_L is the largest phonon frequency. Note that this quantity is in fact just $\Delta U_{\text{ph}} - TS$ and is of high importance when considering the stability of crystals. It is similar to the Gibbs free energy ($G = H - TS = U + PV - TS$) but considering the volume to be constant rather than the pressure. When considering the stability of crystals, it can therefore be seen as a close approximation to the Gibbs free energy, ruling the stability at constant pressure and temperature. Again it only includes the vibrational part.

Specific heat

Finally the specific heat can be computed within the harmonic approximation as,

$$C_v = 3nNk_B \int_0^{\omega_L} \left(\frac{\hbar\omega}{2k_BT}\right)^2 \operatorname{csch}^2\left(\frac{\hbar\omega}{2k_BT}\right) g(\omega)d\omega \tag{1.5}$$

where k_B is the Boltzmann constant and ω_L is the largest phonon frequency. It represents the amount of vibrational energy to be added at constant volume to increase to temperature by 1 Kelvin.

1.3 Machine learning

This section refreshes very briefly some basic concepts in machine learning that are important for the upcoming chapters.

1.3.1 Nuts and bolds

The present work will globally use two types of learning: supervised and unsupervised learning.

Unsupervised learning tackles the task of identifying patterns in unlabeled data. It does so by clustering the data into similar peers or projecting high dimensional data into a smaller subspace.

In *supervised* learning, one wants to map a set $\mathbf{X}$ to a *target* y, given some unknown relation $y = f(\mathbf{X})$. The set $\mathbf{X}$ is called the feature space while an element x from it is commonly called *feature*, *descriptor* or simply input. In other words, the idea is to find some approximate function $\hat{f}$, such that the difference between the predicted value $\hat{y} = \hat{f}(\mathbf{X})$ and the true value $y = f(\mathbf{X})$ is minimal. This discrepancy between y and $\hat{y}$ is commonly assessed by a *loss function* $J(y, \hat{y})$. If y is discrete, this is called a *classification* problem, and if y is continuous this is called a *regression* problem.

In order to do so, many samples (x, y) are given, and are known as the *training set*. Typically, a supervised ML-algorithm will have many degrees of freedom (i.e. *weights*) that will be iteratively tuned in order to minimize the loss function J. Parameters that are not optimized during training, but rather fixed *a priori* are called *hyperparameters*. Many different algorithms exist that try to model a function f and two examples are given in the next section.

Once a model is trained, it is commonly validated on a *test* set, consisting of unknown samples (x, y), i.e. they were excluded from the training phase. This enables one to find out how well a model generalizes to new data. A typical validation-strategy is the *k-fold cross-validation*. This consists in splitting the data in k folds (i.e. subgroups), training the model on $k - 1$ folds and then assess the model on the remaining single fold. This is repeated k-times, such that each fold forms the validation set at one iteration. Finally, the mean error is taken over all folds. Compared to a single validation set, the k-fold cross-validation has the advantage to reduce the variance on the estimate. A schematic representation of a 10-fold cross-validation is represented in figure 1.3.1.

The name *validation* is used (as opposed to *test*) as it is used to find out which model (and its hyperparameters) is best. In this sense, the validation error is biased towards the best model. In order to have an unbiased estimate of the final performance, an independent *test* set is used, unseen during training *and* validation.

When training a ML-model, it is sometimes found that a model performs very well on the training data but much worse on the validation (or test) set. This phenomenon is called *overfitting*. It is often due to a model that is too complex with respect to the number of training samples, and the model just acts as a databank. Several techniques exist to limit this such as *regularization*, *weight pruning*, *bagging*, etc. The inverse phenomenon is called *underfitting*. A more detailed discussion about this will be done in Chapter 5, section 2.

Figure 1.3.1: Schematic illustration of a 10-fold cross-validation strategy. A model is iteratively trained on the 9 folds while tested on the remaining fold. The final error is the mean over the 10 folds. Image adapted from [12].

1.3.2 Learning algorithms

This section does not aim at providing an exhaustive list of all available models but rather at giving the reader two brief examples of regression algorithms. I chosen to present the Random Forest (RF) and the feed-forward artificial neural network (ANN) as they will be used in this work.

1.3.3 Random Forest

The random forest is a simple model to understand that is often used in the literature (at least in the material science field) thanks to its simplicity: it is non-parametric and requires very few optimization. Random Forests are in fact both classification and regression algorithms.

The RF is an ensemble technique in which several decision trees are trained and averaged over their predictions. A decision tree basically partitions the data space at each of its nodes into two subsets. Each node can be seen as a question on the features that eliminates some possible targets. This is repeated until only one possible target remains, referred as the leaf of the tree, which forms the predicted value.[1]

Concerning the algorithm that partitions the space, various implementations exists but they are generally based on reducing the information entropy of the subsets. For more information see Refs. [13] and [13] An example of a decision tree applied on material property prediction is given in figure 2.1.1, Chapter 2, where Heusler compounds are identified.

The problem with decision trees is that they often suffer from overfitting. In order to reduce this, the Random Forest algorithm trains a multitude of smaller decision trees, each on a random training subset (or bootstrap sample) and a random feature subset, hence the name *Random* Forest. In this sense, it generates a bunch of weak classifiers that taken together increases the validation accuracy.

[1]It is in fact possible to have multiple samples at the leaves, and in this case the mean is taken as output value.

1.3.4 Artificial Neural Networks

Artificial Neural Networks (ANN) were initially inspired by the human brain, especially by the neuron cell and the synapse. They are based on the concept of a 'neuron', which basically takes a vector as input, performs a linear combination on it, followed by the application of a non-linear function, called activation function. These neurons are then assembled in layers, where the output from one layer forms the input of the next layer. In this sense they form a directed weighted graph. A schematic illustration of an ANN is given in figure 1.3.2.

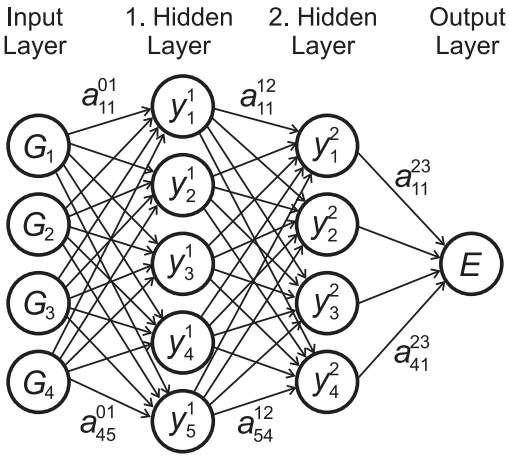

Figure 1.3.2: Schematic illustration of ANN consiting in 2 hidden layers. The $\{G_i\}$ forms the input vector, wile E forms the output value. The fitting parameters a_{ij}^{kl} are represented by the arrows. For clarity, the bias nodes are not drawn. Figure taken from Ref. [14].

The following notations are introduced: the weight a_{ij}^{kl} is connecting neuron i in layer k with neuron j in layer l. The input layer correspond to the subscript 0. Additionally, each neuron i in layer j has a bias term b_j^i. From this any ANN can mathematically by written as:

$$y_i^j = f_i^j \left(b_i^j + \sum_k a_{ki}^{j-1,j} \cdot y_k^{j-1} \right) \tag{1.6}$$

where y_i^j is the output of neuron i in layer j and $f_i^j(\cdot)$ a non-linear activation function. It is thanks to this non-linearity that the arbitrarily functions can be fitted. This latter formula shows clearly how the different layers are interconnected.

In order to choose the values for the weights a_{ij}^{kl} and the biases b_j^i, a loss function is optimized with respect to these weights. For example, the mean absolute error loss function is written as:

$$J = \frac{1}{N} \sum_{i=1}^{n} \|\hat{y}_i - y_i\| \tag{1.7}$$

where N is the number of training samples, $\hat{y}_i$ the predicted value for sample i and y_i the actual value for this sample. In order to optimize this function a gradient descent approach

is typically followed, where each weight a_{ij}^{kl} is iterative updated at time $t+1$ as,

$$a_{ij}^{kl}(t+1) = a_{ij}^{kl}(t) - \alpha \frac{\partial J}{\partial a_{ij}^{kl}} \tag{1.8}$$

where α is called the *learning rate*. The weights are usually initiated by sampling a Gaussian distribution. It can in fact be shown that this latter equation corresponds to the derivative of the error with respect to the output fed in the neural network in the opposite side, extracted at the corresponding weights (the last layer's weights are moreover replaced by the actual value of the corresponding neurons). This can be easily proven by the chain rule and is commonly referred as *backpropagation*.

Another option is to use a mini-batch stochastic gradient descent approach where samples are passed in several subgroups through the network one after another, called a batch, instead of passing all training samples at once. The ANN is sequentially optimized following Eq. 1.8 on each batch. The term epoch is used to denote the number of times all batches have been passed through the network. The advantage of this technique is that it requires less memory and thanks to the stochastic nature it often improves convergence speed for non-convex optimization problems [15].

Finally, ANNs mainly differ by playing on the architecture, i.e. the number of layers and neurons, activation functions, optimizer, loss function, regularization among others. Other types of layers exist too, such as convolutional layers, which form the basis of *deep* neural networks. More information about ANNs and deep learning can be found in the excellent review given by LeCun *et al.* in Ref. [16].

1.4 Conclusion

This chapter introduced some basic concepts in Machine Learning and showed how it could be helpful for the materiel scientist. In particular, providing a fast and effective way for predicting vibrational thermal data for crystalline compounds is really crucial. Indeed, these latter properties are paramount for an accurate estimation of a compound's stability. However, direct methods based on first-principles are often computationally too expensive, making fast screening impossible. Therefore, the upcoming chapters will have as the main goal to provide a fast and effective method for thermal data prediction on crystalline compounds. Last but not least, ML offers much more than a simple regression framework. It enables one to understand and grasp new insights into patterns and relations as will be seen numerous times during this work.

Chapter 2

A brief literature study

This chapter and the remaining work will focus on how machine learning may help the discovery of new compounds, by relying on models that relate system descriptors to desirable properties, especially thermodynamic properties. As explained before, the field of molecular (primarily pharmaceutical and medicinal) science have experienced so far a stronger degree of uptake in machine learning compared to crystalline solids, partially due to the difficulty of representing crystals into a ML-suitable format and the high applicability of molecular science in the pharmaceutical industry [6], [7]. Nevertheless, the application of machine learning for functional materials is an emerging field. One of the first reports applying artificial intelligence to material property prediction appeared in 1998, targeted on magnetic and optoelectronic materials [17]. However, the number of studies really started to increase significantly only since 2010 [18]–[20].

Most of these reports rely on a case per case study, targeted on a specific group of materials (by limiting the training data) and a specific property, with corresponding specific descriptors. Only very recently, more general models are being developed that are applicable on various materials and properties.

This chapter will therefore briefly discuss these different models, by dividing them in three categories. The first section describes the more common ad-hoc methods, that vary broadly in their algorithms and purposes. The second section is more focused on thermodynamic properties, key target of this study, while the third and last section describes the more recent and general models.

2.1 'Ad-hoc' methods

Ad-hoc methods are machine learning algorithms that have a specific purpose in mind on a particular type of crystal structure, e.g predicting the band gap of a certain class of materials. Typically, hand crafted descriptors are tailored in order to suite the physics and are the major point of attention, while common simple-to-use ML models are chosen. They form the major part of the literature. Learning is commonly performed on experimental data, or on theoretically computed databases (see chapter 1). There are two reasons why these ad-hoc methods are so popular. First, and as written before, it is due to the difficulty of representing crystalline solids in an effective way that is easily understandable by the statistical learning procedure. Developing flexible and transferable representations is one of the most important research areas in machine learning for crystalline solids [6]. Secondly,

due to the limiting number of samples for many problems, much better performance is achieved by restricting on a particular structure, which is therefore inherently built into the model.

Finally, it is important to note that most published ML works can mainly be differentiated and analyzed by three key contributions: descriptors, ML model and dataset. The latter often restricts the generalization space and thus the extent of applicability to new materials.

A first example concerns the identification of Heusler compounds, which are intermetallics exhibiting diverse physical properties attractive for applications in thermoelectric and spintronic materials. They are difficult if not impossible to detect by standard diffraction techniques. Therefore, Oliynyk *et al.* proposed a machine learning model that identifies Heusler compounds of the type AB_2C [21]. A random forest is used as the ML-model, while various compound-only descriptors are used, involving the group number of element B, the total number of p-electrons or the radius difference between species A and B. It was then trained on 1948 samples from experimental data. It showed a remarkable true positive rate of 0.94 and a false positive rate of 0.01. An example of a simplified decision tree for this classification problem is showed in figure 2.1.1. In particular, novel predicted candidates MRu_2Ga and RuM_2Ga were synthesized and confirmed to be Heusler compounds. Moreover, it shows how fast ML-based predictions can be compared to exact calculation by taking only 45 min to make the full set of predictions on over 400,000 candidates, that is, less than 0.01 seconds per compound. Finally, this model illustrates well the 'ad-hoc'-method as it is restricted to AB_2C-compounds and uses a well-selected specific set of features known to be determining for Heusler compounds.

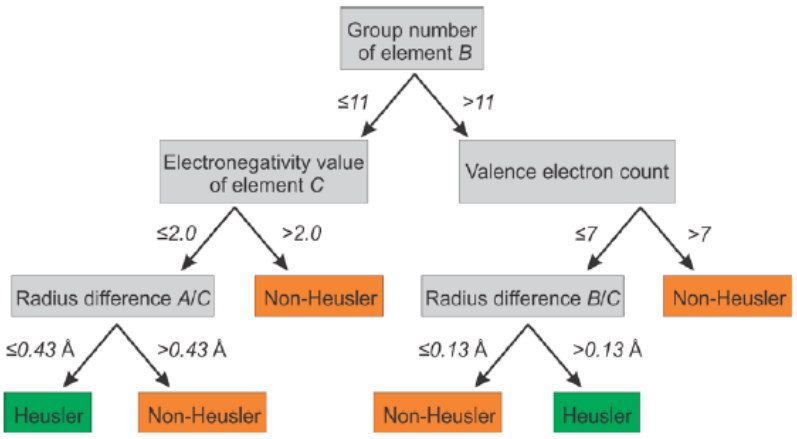

Figure 2.1.1: Illustration of a single decision tree, trained only on a fraction of the descriptors, for the prediction of Heusler structures for candidates of the type AB_2C, as presented in the work of Oliynyk *et al.* [21].

Another example, concerning force field fitting by machine learning, was proposed by Behler in 2011 [14]. Force field fitting is mainly concerned by finding potential energy surfaces of a system of (bounded) atoms, i.e. the potential energy of the system as a function of the atomic coordinates. This enables one to compute forces and perform molecular dynamics. Unfortunately, computing these forces exactly by ab-initio calculations is computationally intensive. Therefore, a first alternative is to fit the potential by a physically-motivated

functional, which is appealing and often offers a good transferability but however generally lacks in numerical accuracy. To solve this, an alternative approach is to fit the potential by a pure mathematical function such as splines, Taylor expansions or Gaussians. They have a good numerical precision but are usually less transferable. Another example of this latter class of functions are Neural Networks (NN), which additionally have the tendency to be more transferable then simpler mathematical functions.

The major problem when fitting potentials with machine learning models is to find a good feature vector that represents the system coordinates. Simply using the Cartesian coordinates is not a good choice as they are not rotationally invariant while the potential has this latter property. Therefore, Behler introduced the concept of many-body symmetry functions which basically capture the local environment of an atom over a certain cut-off radius. For example, the discrete integration of the product of a Gaussian function and the radial distribution function over a restricted window forms the second symmetry function:

$$G_i^2 = \sum_j e^{-\eta(R_{ij}-R_s)^2} \cdot f_c\left(R_{ij}\right) \tag{2.1}$$

where R_{ij} is the distance between the site i and every neighbor j, R_s a suitable shift distance, η a parameter ruling the width of the Gaussian and finally f_c a cut-off function. For an exact definition of this latter function and other symmetry functions, the reader is referred to Chapter 3 (and especially figure 3.3.7 for an illustration of the symmetry functions), which contains more detail about the present features. From these features, a neural network is trained per atomic species (i.e. element) that maps the symmetry functions to an energy contribution. This NN per element strategy is needed as the central species is not encoded explicitly in the symmetry function, but rather implicitly encoded in the NN. Moreover, it is computationally better to have n independent NNs than n interconnected NNs. The contribution from the different species are finally added to yield the potential energy, as depicted in figure 2.1.2.

Another benefit of using these symmetry functions is that they are continuous with respect to the atomic positions, which guarantees derivability, a necessary condition to compute force fields. Note that the same is true for the NN model.

Concerning formation energies, Rupp *et al.* proposed a framework to predict atomization energies for small molecules [22]. Any molecule is first transformed into a Coulomb-matrix (see Eqs. (3.10) and (3.11) for an exact definition) which represents the Coulomb interaction between species and is therefore both a spatial and elemental representation. This matrix is flattened by computing the corresponding eigenvalues, which form the feature vector. From this, the distance $d(\mathbf{M}, \mathbf{M}')$ between two molecules is defined as the euclidean norm of the difference between both feature vectors: $d\left(\mathbf{M}, \mathbf{M}'\right) = \sqrt{\sum_I \left\|\epsilon_I - \epsilon'_I\right\|^2}$, where ϵ represent the previously mentioned eigenvalues in order of decreasing absolute value. For matrices of different dimensionality, the smallest vector is extended by zeros. Finally, atomization energies are predicted, as a sum over weighted Gaussians:

$$E^{est}(\mathbf{M}) = \sum_{i=1}^{N} \alpha_i \exp\left[-\frac{1}{2\sigma^2} d\left(\mathbf{M}, \mathbf{M}_i\right)^2\right] \tag{2.2}$$

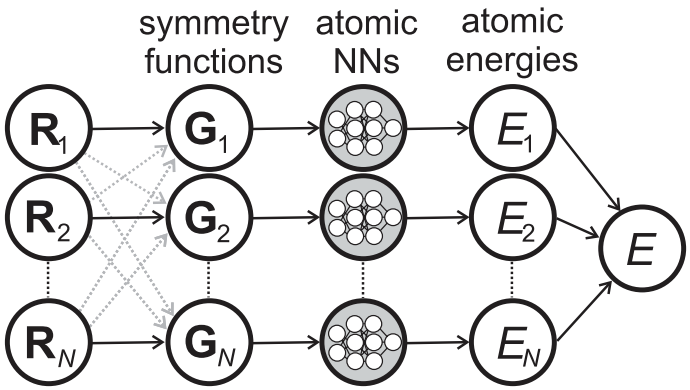

Figure 2.1.2: Schematic representation of the work proposed by Behler [14], a neural network in order to represent high dimensional ab-initio-potential-energy surfaces. Each line represents an atom at coordinates $\mathbf{R}_i$. These coordinates are transformed into a set of symmetry-function values $\mathbf{G}_i$, taking the local environment into account (itself depending on neighbors coordinates as indicated by the dotted arrows). The symmetry-function values represent the input vectors for atomic NNs yielding the atomic energy contributions E_i . The total energy E is the sum over all E_i. Image taken from Ref. [14].

where i runs over all molecules $\mathbf{M}_i$ in the training set. The parameter σ is fixed a priori (and is thus a hyperparameter), while regression coefficients $\{\alpha_i\}$ are set in order to minimize the squared training error in addition to a L^2-regularization. This is nothing but a kernel ridge regression with a Gaussian kernel. Concerning the accuracy, when training on more than 1000 molecules, the ML model becomes competitive with mean-field electronic structure theory, at a fraction of the computational cost. It is remarkable to see how this ML model can be seen as an approximation to the time-independent Schrödinger equation, $H\Psi = E\Psi$, where the Hamiltonian, defined by a set of nuclear charges $\{Z_I\}$ and atomic positions $\{\mathbf{R}_I\}$, is replaced by the Coulomb-matrix. In other words, the ground-state potential energy found by solving the Schrödinger equation, $H\left(\{Z_I, \mathbf{R}_I\}\right) \overset{\Psi}{\longmapsto} E$, can be replaced by a ML-approach: $\{Z_I, \mathbf{R}_I\} \overset{\text{ML}}{\longrightarrow} E$.

This framework could be easily transferred to solid crystals by taking the unit cell as the molecule in the previous algorithm, which works remarkably quite well. However, this does not take into account long-range Coulomb interactions over many cells in the lattice. Therefore, extensions to periodic crystals have been proposed, such as the sine-Coulomb matrix presented in Chapter 3.

A more recent example involves the identification of lattice symmetries. Classic space-group-determination algorithms are based on identifying the compatible symmetry operation and then compare them with all possible space groups to obtain the correct label, as done for example in the FINDSYM (see Ref. [23]) package. This works for idealized crystal structures, but whenever some significant perturbation to the positions is introduced - as a consequence of intrinsic defects, impurities or experimental noise - these algorithms fail. Therefore, Ziletti *et al.* introduced a classification algorithm using deep learning, see figure 2.1.3 [24]. The crystal structure is first transformed into an image, by simulating the X-ray diffraction intensity formed by an incident plane wave. This image is then passed through a convolutional neural network (CNN) which classifies the crystals following their (average)

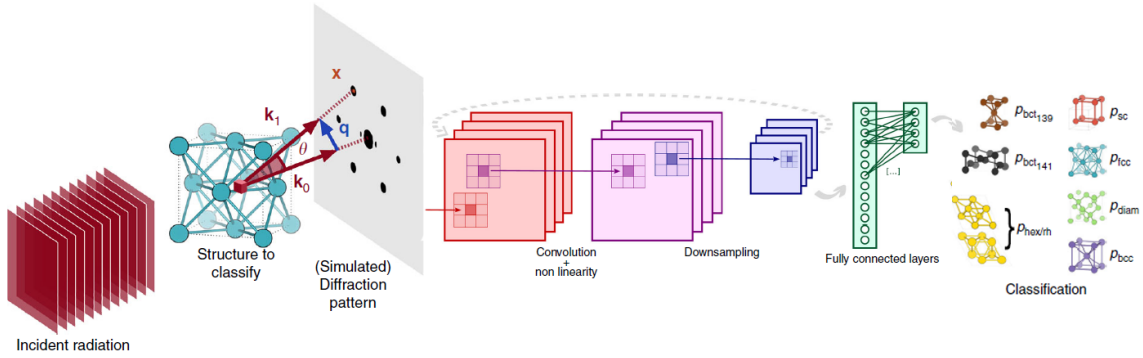

Figure 2.1.3: Schematic representation of the work proposed by Ziletti *et al.* [24], where the 'mean' space group of a crystal is predicted using a convolutional neural network (CNN). The crystal structure is first transformed into an image, by computing the diffraction pattern intensity caused by a plane wave. Then, the CNN predicts a probability per space-group class. Image adapted from Ref. [24].

space group. The output forms a vector giving the probability to belong to a particular class. When trained, through backpropagation and in order to minimize the prediction error on a given labeled training set, it is seen how the first layer kernels (or filters) are very similar to diffraction patterns, with bright peaks, while going deeper reveals more complex patterns. The final algorithm shows a 100% accuracy for random perturbation up to 0.06 in standard deviation and vacancies up to 25%.

To finish, we briefly mention other examples. Hautier *et al.* used a machine learning model based on experimental data in order to predict possible novel ternary oxides, which are then simulated by ab-initio computations to confirm stability [20]. Saad *et al.* applied unsupervised learning to binary compounds, by for example grouping compounds into characteristically-similar peers [25]. Zeni *et al.* used Gaussian process regression in order to fit force fields for metallic nanoclusters [26]. Kolb *et al.* proposed an open-source framework, named PROPhet, based on neural networks to predict a variety of properties including band gap energies [27]. It also offers integration with first-principle methods by for example giving the ability to train charge-density functionals to arbitrary system properties. Concerning magnetic properties, Lam Pham *et al.* developed a novel way to represent materials named "orbital-field matrix (OFM)" based on the distribution of the valence shell electrons [28]. By using the OFM with a random forest they predicted the DFT-computed magnetic moment for lanthanide-transition metal alloys, with an MAE of 0.05 μ_B. Finally, Stanev *et al.* tackled the discovery of high T_c superconductors by proposing a model that predicts the critical temperature based on the composition alone [29]. More precisely, a random forest is used together with the Magpie elemental features on the SuperCon database [30], [31]. Then the Inorganic Crystallographic Structure Database (ICSD) was screened for new superconductor discovery. In particular, 35 non-cuprate and non-iron-based oxides were identified as candidate materials.

Only a few examples out of many were given here, and more and more articles, perspectives, and reviews are nowadays released in the literature, as the application of ML techniques to material problems is relatively recent. Therefore, the reader is referred to the excellent works given in Refs. [6] and [2] for more examples that illustrate the applicability of ML in material science.

2.2 Thermodynamic

It was already stated that a fast and reliable way of characterizing and predicting thermodynamical properties of materials is clearly needed. For instance, the vibrational entropy may easily alter the ordering of the energies of the structures and modify which one is the most stable as the temperature is raised. However, while ab-initio relaxed crystal structures and groundstate energies are routinely provided in the HT databases, the thermal properties are much scarcer, as computing inter-atomic force constants is very demanding.

With respect to the latter, this section will present the (few) works that tried to tackle the problem of predicting vibrational properties of solids using ML, and their current limitations.

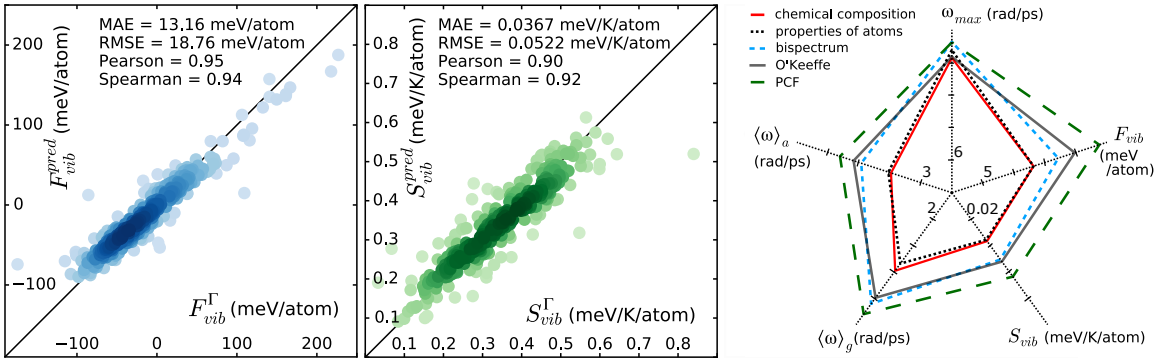

Figure 2.2.1: Summary of the main results of Legrain *et al.* [32], where the vibrational entropies and free energies are predicted from the composition of crystals only. *Left*: ML-predicted *vibrational free energy* with respect to the the DFT-computed value. *Middle*: ML-predicted *vibrational entropy* with respect to the the DFT-computed value. *Right*: Comparison in performance for the different proposed descriptors. Images taken from the original work, see Ref. [32].

A first work was published in 2017 by Legrain *et al.*, predicting vibrational free energies and vibrational entropies for inorganic crystals at 300K. The dataset consists of 292 inorganic crystals. They were computed by randomly selecting 600 compounds from the Inorganic Crystal Structure Database (ICSD) section of the aflow.org repositories (52671 compounds), where compounds containing noble gases or element with atomic number higher than 112 were removed. Then a high-troughput method computed the phonon frequencies, from where 292 samples were kept (samples that did not run smoothly or resulted in imaginary frequencies were discarded). Concerning the descriptors, a composition-only approach was followed. The input vector v was simply formed by assigning the molar fraction of element i to v_i. As the dataset included 87 elemental species, the feature vector is of the same length. For instance, Mg_2Si is described by $\left(0, 0, \ldots, 0, \frac{2}{3}, 0, \frac{1}{3}, 0, \ldots, 0\right)$. Concerning the model, a random forest containing 500 trees is used or a support vector machine (SVM) with a Gaussian kernel, depending on which model gave the best performance. To assess performance of the ML method, a k-fold cross validation scheme is employed with $k = 514$, and the performance was averaged over all (i.e., 10) cross validations. A mean absolute error (MAE) of 0.037 meV/atom (15 meV/atom) is achieved for the vibrationnal entropy (vibrationnal free energy), whose values range from 0 to 0.9 meV/K/atom (from 100 to 200 meV/atom). This is quite remarkable given the simplicity of the method. The article reports that no improvement were found by using more complex descriptors, such as

Bartok-Partay's Bispectrum Components or O'Keeffe's Solid Angles (see the original article for a detailed definition of those descriptors), which is not surprising given the fact that a rather small training set was used. However, one can expect that for larger databases (exceeding one thousand compounds) more complex models would certainly be beneficial. Moreover, while this method is good for fast screening, the accuracy is not as high as one would like. The results of this work are summarized in figure 2.2.1.

To raise the corner of the veil, the present work will be able to predict the same properties but with a 50 % lower error.

Further work of Legrain *et al.* predicted IFCs directly from tensorial descriptors, depending only on the species and distance between atom pairs, using a random forest. It has the advantage that further quantities (such as phonon densities or vibrational entropies) can directly be deduced from the predicted IFCs.

The interatomic force constants between atom i and j constitute a second-order tensor defined by the second derivatives of the energy E with respect to atomic displacements:

$$\Phi_{ij} = \left(\nabla_{\mathbf{r}_i} \otimes \nabla_{\mathbf{r}_j}\right) E \tag{2.3}$$

The tensorial discriptors used for predicting these IFCs, represented by $\mathbf{D}_{ij}^{(2)\alpha}$, are second order derivatives of scalar descriptors that represents the local environment (such as for force field fitting) between two atoms i and j:

$$\mathbf{D}_{ij}^{(2)\alpha} = \left(\nabla_{\mathbf{r}_i} \otimes \nabla_{\mathbf{r}_j}\right) g_{ij}^{\alpha} \quad \propto \quad g_{ij}^{\alpha} \mathbf{r}_{ij} \otimes \mathbf{r}_{ji} \tag{2.4}$$

where $g_{ij}^{\alpha} = e^{-\left(\frac{\mathbf{r}_{ij}}{a_\alpha}\right)^2}$ (accounting for the local environment) and $\{a_\alpha\}$ a set of radii spanning a few interatomic distances. They transform as rank-2 tensors and are invariant with respect to rotations, symmetries of the system, as well as permutations among atoms of the same species. These descriptors[1] are then used to fit the IFCs:

$$D_{i,j}^{(2)\alpha} \overset{\text{ML}}{\Longrightarrow} \Phi_{i \neq j} \tag{2.5}$$

The ML-model is a random forest consisting of 100 trees. The particularity of the work is that it mainly focuses on polymorphs, i.e. a compound of fixed composition that adopts different structures. In particular they trained the model on 121 metastable structures of $KZnF_3$. For this dataset, the MAE on a 10-fold cross validation is $0.17\,\mathrm{eV/\mathring{A}^2}$. This corresponds to an MAE of 0.009 meV/K/atom (2.92 meV/atom) on the vibrationnal entropy (free energy) at 300K when computing these quantities explicitly from the IFCs. Knowing that most entropies (free enthalpies) for the $KZnF_3$ polymorphs are situated between 0.25 and 0.35 meV/K/atom (-20 and 10 meV/atom), these results are a little disappointing. It shows that even if the force constant are rather accurate, errors accumulate when computing derived properties, and could still violate conservation rules of the system leading to unphysical results. Spurious imaginary phonon frequencies even appear in some of the compounds.

[1] The actual descriptors are in fact slightly different by accounting for periodicity and chemical species, but are omitted here for simplicity. See the original paper for more details.

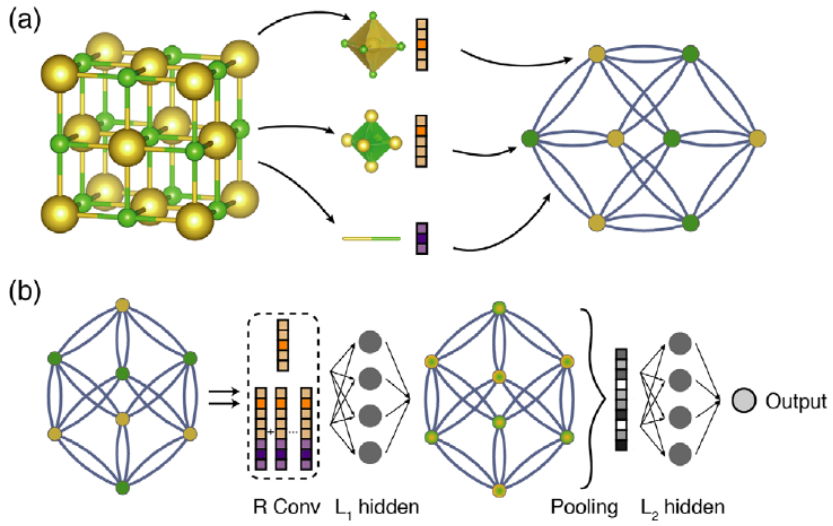

Figure 2.3.1: Illustration of the crystal graph convolutional neural network (CGCNN). (a) Construction of the crystal graph. Crystals are converted to graphs with nodes representing atoms in the unit cell and edges representing atom connections. Nodes and edges are characterized by vectors corresponding to the atoms and bonds in the crystal, respectively. (b) Structure of the convolutional neural network on top of the crystal graph. R convolutional layers and L_1 hidden layers are built on top of each node, resulting in a new graph with each node representing the local environment of each atom. After pooling, a vector representing the entire crystal is connected to L_2 hidden layers, followed by the output layer to provide the prediction. Taken from Ref. [33]

The main limitation is that this work is restricted to polymorph structures, with no guarantees when generalizing over different classes and is rather limited in accuracy when computing derived quantities due to potential accumulation of errors. Nevertheless, it has the advantage of predicting IFCs with a good accuracy, and derived properties can be easily computed for a first on-the-fly estimation.

2.3 Advanced models

The presented models in this section are labeled 'advanced' in the sense that they are very flexible and general. They are made with no specific purpose in mind and can therefore be trained on any given dataset. Different classes of materials or a mixture of classes can be used, with the desired targets. They are intrinsically more complicated and often based on a deep-learning approach. Thanks to their flexibility they are able to extract the relevant features by themselves from a very raw material representation (e.g. CIF-file). Of course there is downside to these type of models, they are namely computationally expensive to learn and require often more than 10,000 samples to be effective.

An example of such a model is the Crystal Graph Convolutional Neural Network (CGCNN) proposed by Xie *et al.* [33]. It transforms a crystal structure into a graph whereafter a convolutional neural network automatically extracts representations that are optimum for predicting target properties. This framework is schematically depicted in figure 2.3.1, and will be discussed next. A structure is first given as input, giving the species and atomic positions in the unit cell. This is transformed into an undirected multigraph where atoms

become vertices and bonds become edges. It is a *multi*graph as it is possible to have multiple edges between a same pair of vertices, due to the periodicity of the crystal structure. The next step is to assign a vector $\boldsymbol{v}_i$ to each vertex i (atom) and a vector $\boldsymbol{u}(i,j)_k$ to each edge (bond), corresponding to the k-th bond between atom i and atom j. A choice for $\boldsymbol{v}$ could be the vector formed by zeros everywhere except at the atomic number where a 1 is put. A similar choice can be taken for $\boldsymbol{u}$, but based on the bond length. These first two steps are represented in figure 2.3.1 (a).

After this, a convolution is performed on each vertex by taking its chemical environment into account. More precisely each vector $\boldsymbol{v}_i$ is updated as follows:

$$\boldsymbol{v}_i^{(t+1)} = \boldsymbol{v}_i^{(t)} + \sum_{j,k} \sigma\left(\boldsymbol{z}_{(i,j)_k}^{(t)} \boldsymbol{W}_f^{(t)} + \boldsymbol{b}_f^{(t)}\right) \odot g\left(\boldsymbol{z}_{(i,j)_k}^{(t)} \boldsymbol{W}_s^{(t)} + \boldsymbol{b}_s^{(t)}\right) \tag{2.6}$$

with $\boldsymbol{z}_{(i,j)_k}^{(t)} = \boldsymbol{v}_i^{(t)} \oplus \boldsymbol{v}_j^{(t)} \oplus \boldsymbol{u}_{(i,j)_k}$.
Here $\oplus$ represents concatenation, $\odot$ represents element-wise multiplication, $g(\cdot)$ is any non-linear activation function while $\sigma(\cdot)$ represents a sigmoid function. The $\boldsymbol{W}$-matrices are learnable weights. Intuitively, $\boldsymbol{W}_f$ and $\boldsymbol{W}_s$ can be seen as two convolution kernels that take into account the environment of atom i, while $g(\cdot)$ and $\sigma(\cdot)$ introduce some non-linearities between convolution layers. In fine, one obtains for each atom i (or vertex) a new representation characterized by a series of vectors $\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, ..., \boldsymbol{v}_i^R$, which is represented by the second graph in figure 2.3.1 (b).

A pooling layer is then used to produce a single overall feature vector $\boldsymbol{v}_c$ for the crystal, which can be represented by a pooling function,

$$\boldsymbol{v}_c = \text{Pool}\left(\boldsymbol{v}_0^{(0)}, \boldsymbol{v}_1^{(0)}, \ldots, \boldsymbol{v}_N^{(0)}, \ldots, \boldsymbol{v}_N^{(R)}\right), \tag{2.7}$$

which satisfies permutational invariance with respect to atom indexing and size invariance with respect to the unit cell choice. The original work uses a normalized element-wise summation function for simplicity, and seems to work well.

Finally, from $\boldsymbol{v}_c$ two hidden fully connected layers predict the output target. All weights (including those for the convolution) are trained in order to minimize the prediction error (i.e. difference between the predicted value and the 'true' labeled value), by using backpropagation and stochastic gradient descent (SGD).

The CGCNN was trained on 8 different DFT calculated properties, with $\sim 10^4$ crystals per property having various structure types and compositions. The MAE of the CGCNN predictions wrt. the DFT values was in the same order than one can expect from DFT compared to experimental data, which shows the high accuracy of the model. However, it systematically requires huge datasets of at least 10^4 samples.

Finally, the model was shown to be interpretable in some cases. In short, by replacing the last hidden layer by a linear function, one can easily find the contribution of each atom (and its environment, represented by $\boldsymbol{v}_c$) to the final outputted value. This enables for example to find the mean contribution of each elemental species to the energy above hull data of perovskites as illustrated in figure 3 of the original paper. This information can be utilized to discover empirical rules for materials design and in general, chemical insights gained from CGCNN can significantly reduce the search space for high throughput screening.

Another interesting model is the one proposed by Verpoort *et al.* [34]. Their idea is to implement a function f that satisfies the fixed point equation:

$$f(\mathbf{x}) \equiv \mathbf{x}, \tag{2.8}$$

where x is a vector containing a list of properties. It considers all properties, i.e. defining properties (such as composition and structure) and physical properties (such as thermal conductivity or yield stress), on the same ground. Both categories form the input and output vector. However some inputs can be missing and therefore the identity function, $f(\mathbf{x}) = \mathbf{x}$ is not a viable solution. It should in fact be orthogonal to the identity operation and the adopted solution is therefore a neural network, with one hidden layer and a *tanh* activation function, that tries to find the correct value of the missing entries by relying on other *given* entries. This is forced by setting the weights in the hidden layer that connects same properties from the input layer to the output layer to zero.

The model is then trained in order to minimize the loss function,

$$J = \sqrt{\frac{1}{N} \sum_{\mathbf{x} \in X} [f(\mathbf{x}) - \mathbf{x}]^2}, \tag{2.9}$$

following a stochastic gradient descent on a given training set, where X represents the training set.[2] Furthermore, a whole suite of ANNs are trained simultaneously, and the returned value is the mean over this suite. This enables moreover to compute a standard deviation and expected uncertainty, represented by $f^\sigma(\mathbf{x})$.

In order to predict missing properties, an iterative approach is followed by re-injecting temporary outputs again as inputs until convergence. This procedure is depicted in figure 2.3.2.

Finally, the ANN can be used to search for erroneous entries in a data set. Intuitively, if the predicted quantity is substantially away from the given quantity, it could potentially be erroneous. In practice, the following quantity is computed:

$$\Delta_\sigma(\mathbf{x}) = \frac{f(\mathbf{x}) - \mathbf{x}}{f^\sigma(\mathbf{x})}, \tag{2.10}$$

which represents the number of standard deviations that this entry lies away from the ANN's prediction. If this value is higher then an upper limit of Δ_σ, the corresponding entry is considered erroneous. For more details, the original paper discusses how a reasonable upper limit can be chosen (as well as an iterative approach to follow when removing such entries, as the model should be retrained after removing them). As main result they applied this error detection scheme to two commercial databases: MaterialUniverse and Prospector Plastics, and were able to find 20 errors. These errors were confirmed by looking back into the primary data sources, and were thus transcription errors from that primary data set into the database.

[2]The actual loss function is in fact slightly different, as it additionally accounts for *functional properties*, e.g. properties at multiple temperatures. It is omitted here for simplicity, and does not alter the main idea. For more about this, see the original paper, Ref. [34].

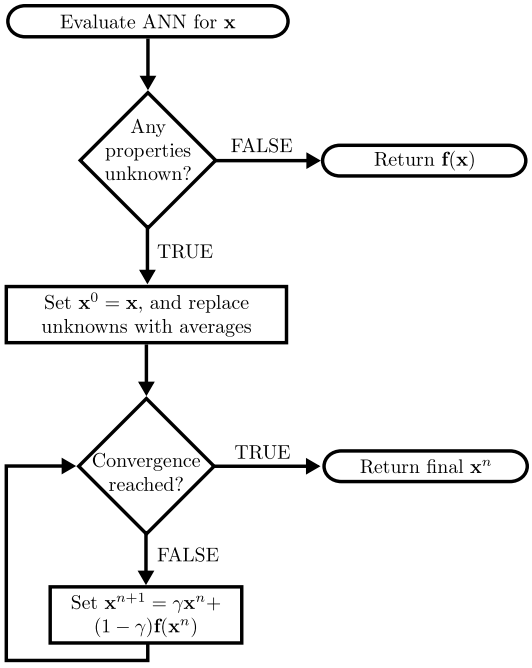

Figure 2.3.2: Schematic representation on how to impute missing data in a property entry $\mathbf{x}$. The missing values are first replaced by averages and are then iteratively replaced until convergence. Image taken from Ref. [34].

2.4 Conclusion

This chapter showed the diversity of methods available today. It was seen how most models are often structure and target specific, which intrinsically limits the generalization space. They were therefore labeled as 'ad-hoc'-methods. Recently, more advanced models based on convolutional neural networks (such as the CGCNN) have been published, with promising results. However, they often require too much data than available in order to be accurate enough, at least for vibrational properties. These current limitations can be formulated into the following question.

How can one, as accurate as possible, predict properties on a large scale, given many but limited datasets containing various properties ?

An attempt to answer this question will be given in Chapter 5. But before that, the general problem of converting raw materials into features and gaining insights from this will be discussed in the next two chapters.

Chapter 3

Matminer feature analysis

This chapter introduces *matminer*, a python library for performing data mining in the field of materials science. It contains various routines for (i) obtaining materials properties from various databases, (ii) featurizing complex materials attributes (e.g., composition, crystal structure, band structure) into physically-relevant numerical quantities, and (iii) graphically analyzing the results of data mining. [5]

The first section gives a historical motivation behind matminer. This is followed by a basic description of its functionalities, at the intersection of material science and machine learning. Finally, the different available featurizers are presented with an in-depth physical explanation of the generated features. Therefore, this chapter has the main role of enhancing interpretation and expertise concerning the available features. These features, generated by matminer, are at the heart of this work, as the upcoming predictive models will be based on them.

In particular, those features will be analyzed statistically in the next chapter, in order to find interdependence between them and build a smart feature selection model.

3.1 Motivation

General-purpose methods that automatically predict materials properties based on Machine Learning techniques are based on three major ingredients: a database, a set of attributes generated for each material and a machine learning algorithm. Fortunately material databases are now quite numerous, mainly built from first principles, such as the Materials Project [5] or OQMD [35] containing several thousands of materials. Machine learning models have a relative long history, becoming increasingly popular the last 15 years thanks to growing databases and computational performances. This popularity lead to many high level frameworks, such as the scikit-learn python package. With these two ingredients steadily established, probabilistic materials property learning was quite straightforward. However it required development of the third ingredient, i.e. feature generation.

Initially, this had led to many case per case models, using specific datasets and specific features for a very specific target. Some examples are models for predicting melting temperatures of binary inorganic compounds [36], formation enthalpy of crystalline compounds [37], [38], crystal structure stability at certain compositions [39]–[42], band gap energies of certain classes of crystals [43], [44] and mechanical properties of certain alloys [45], [46]

among others. Those methods demonstrated the promise of machine learning for material science, however they covered only a fraction of the properties and material classes. There was a clear lack of broadly-applicable machine learning models, such as a general band-gap energy prediction for all classes of materials. In particular, routine methods for transforming raw materials to numerical vectors was missing in order to exploit existing machine learning models.

Secondly, most featurization approaches that had been proposed (and are proposed) in the literature do not have an available software code, are moreover not open-source or are distributed across many repositories. This means that reproducing some published methods require a significant time investment, and is thus less accessible.

As a first step towards the solution, Ward *et al.* proposed in 2016 a general-purpose machine learning framework for predicting properties of inorganic materials consisting mainly by 145 elemental property related features [30]. This enabled one to easily featurize a material to a fixed-length numerical vector and subsequently use an adequate predictive model. By using a broad variety of elemental features, very different properties could successfully be predicted, such as band gap energy (principally based on s, p, d and f-electron occupancy) and formation energy (principally based on the difference of electronegativity of the constitual elements). However, the composition alone does not perfectly describe a material. It should be extended to additional features accounting for the structure and local environments, which resulted in additional features published later in Ref. [47]. Convinced that a general-purpose library representing material characteristics could be made, Ward *et al.* came with a python package named *matminer* in 2018. It not only offers the ability to generate a family of descriptors but also gives easy access to material databases and ML methods. Furthermore, matminer proposes a uniform interface, despite the diversity of methods, making it easy for researcher too quickly iterate through data and features in order to determine which one are suited for their application.

3.2 Matminer presentation

Matminer is a python library, offering many useful functions and instances for big data analysis on materials. It is a complete framework in the sense that materials can be first imported, then featurized, and finally visualized all within matminer. As it uses the common *pandas* dataframe representation, it is easy to further process the dataframes with standard libraries such as scikit-learn, which is the de facto standard machine learning library of Python.

This is made possible by integrating different existing libraries such as pymatgen and plotly into one central framework, making material learning easily accessible for new users. The interplay of these different parts and the general and workflow are shown in figure Figure 3.2.1.

Some of the most useful features of matminer are mentioned hereafter.

1. Datasets from a variety of material databases can be easily imported, and are all formatted in pandas dataframes. This makes interaction with the different databases more uniform, with a single consistent API. Supported databases are Citrination [49], the Materials Project [50], Materials Data Facility [51], and Materials Platform for Data Science [52]. In addition, a generic MongoDB interface supports data retrieval

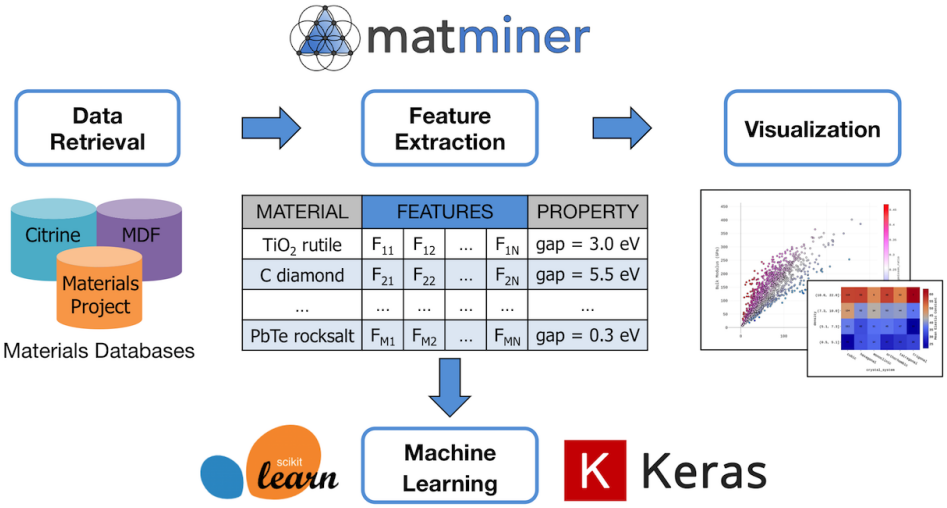

Figure 3.2.1: Main capabilities of matminer. It consists of three main parts: (i) Data retrieval trough a variety of material databases, (ii) feature generation from raw material data and (iii) re-useable and customizable recipes for visualizing materials data. Data is retrieved and processed in a way that makes it simple to integrate matminer with external machine learning libraries such as scikit-learn and Keras. Taken from Ref. [48].

from any MongoDB resource.

2. Materials are transformed to descriptors by a vast amount of featurizers classes. The material structure is for example transformed to the space group number or volume per atom descriptor, which is added as a new column to the pandas dataframe. More details about the different available features are explained in the upcoming section.

3. Results or intermediate computations can be visualized thanks to plotting tools offered by matminer, i.e. the figrecipes module. It uses the Plotly library, which has the advantage of offering interactive plots. Individual material points can therefore be identified by simple mouse hovering.

More information about matminer can be found in the Ref. [48]. The source code is available in Ref. [53].

3.3 Features

Matminer featurizers methods are all regrouped under the `featurizers` module. This module is organized into five submodules: `composition`, `structure`, `site`, `bandstructure` and `dos`. Each of those submodules contains a number of featurizer classes (called featurizers hereafter), for a total of 39 of them. In turn, each of those featurizers (classes) generates a number of actual features (i.e. pandas columns), typically around ten to hundred of features. In total, when running all possible featurizers, one could generate several thousands of different features. This featurizer hierarchy is depicted in figure 3.3.1.

Each featurizer inherits from the `BaseFeaturizer` class, which defines a common template. In particular, it contains the `featurize` method which transforms a single raw data object (e.g. structure) to a list of features or the `featurize_dataframe` function (based on the previous function), that transforms a whole dataframe into descriptors.

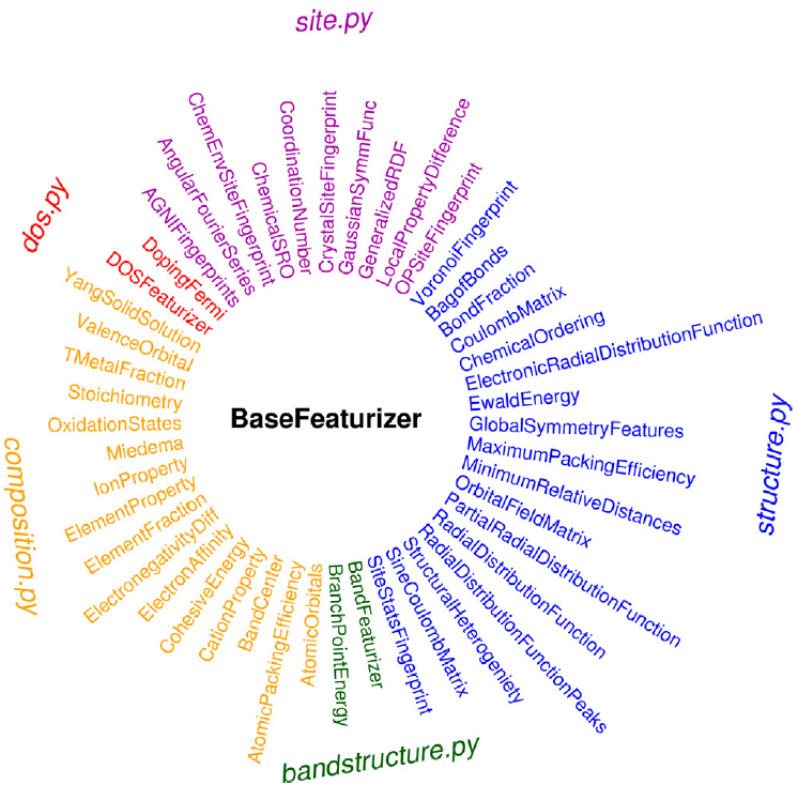

Figure 3.3.1: Overview of the at the time of writing 47 available featurizers in matminer. They are subgrouped in 5 modules: `composition`, `structure`, `site`, `bandstructure` and `dos`. Each featurizer class can produce one to hundreds individual features. Taken from Ref. [48].

In addition to those individual featurizers, matminer offers the `FunctionFeaturizer` function that applies a mathematical function (product, square root, ...) on one or several features in order to generate a large space of possible optimal features.

The feature generation step is a very important step as it directly affects the performance of the machine learning model. It is therefore required to exploit domain-knowledge, i.e. carefully designing and choosing the features convinced by intuition and knowledge. By this motivation, the different available features will be presented, in order to give physical meaning to them. Moreover, it is certainly important to know how these different features are related to each other, again to wisely choose them later on. Therefore, after presenting the features in this chapter, correlations and relations between features will be investigated in the next chapter.

3.3.1 Composition

Composition-based features rely on the composition object, as defined by the `pymatgen` library. This composition object mainly describes which elements are present and in which proportions. Examples of compositional features are the mean atomic number or the mean electronegativity of the constituent elements of the material. Such type of features have been shown to work well on predicting inorganic crystal formation energies [30]. The complete list of different featurizers are detailed below.

ElementFraction. The composition is transformed to a vector V where V_i is set to the molar fraction of element i. In practice, each column represents a chemical element and is set nonzero if it composes the considered compound. It should be noted that it creates a rather large sparse vector, as a compound typically contains only a few elements out of the approximately hundred of possible elements.

ElementProperty, see Refs. [30], [54] and [55]. This class generates a list of various composing elemental properties related statistics. Examples are: the mean melting temperature of the constituent elements in their pure state, the mean row (or column, atomic number) in the periodic table of the constituent elements, mean number of electrons in the s, p, d or f-orbitals of the constituent elements, and many others. Often the standard deviation of the previous quantities are also taken (and other statistics). For a complete list of the elemental properties, the reader is referred to the database that is used to generate them, which is `pymatgen` [54] in the present work.

Stoichiometry, see Ref. [30]. Each composition is first transformed to a vector V containing the elemental fractions V_i of the constituent elements. Then, various L^p norms are computed as features. For example SiO_2 is represented by the vector $(\frac{1}{3}, \frac{2}{3})$, which is subsequently transformed to the feature vector (2,0.745,...), corresponding to the $(L^0, L^2, ...)$-norm of the former vector.

AtomicOrbitals, see Ref. [56]. The highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO) are estimated from the atomic orbital energies of the composition. The feature consists in the HOMO/LUMO character (s, p, d or f) and the corresponding HOMO/LUMO element.

AtomicPackingEfficiency, see Ref. [57]. The first feature is an Atomic Packing efficiency (APE), which is based on the amorphous packing of hard spheres. It measures the difference between the ratio of the radii of the central atom to its neighbors and the ideal ratio of a cluster with the same number of atoms that has optimal packing efficiency. The APE can be positive or negative. The second feature is the L_2 distance between the material and a k-cluster that has a packing efficiency below a certain threshold from ideal (i.e. zero).

BandCenter, see Ref. [58]. This feature computes a rough estimation of the absolute position of the band center. It is found by taking a simple geometric mean of the electronegativities:

$$\prod_i X^{e_i/\Sigma_i} \tag{3.1}$$

where X is the Pauling electronegativity of species i, a_i is the molar fraction of the species i and $\prod_i$ is defined as $\sum_i = \sum_i e_i$.

CohesiveEnergy, see Ref. [59]. The cohesive energy of the crystal is defined as the energy (per unit cell) required to form the crystal starting from the elementary constituents (here taken as the elementary crystals).

For each compound, it is computed as:

$$E_{\text{coh}} = -E_{\text{form}} + a_i \cdot E_{\text{coh,i}} \tag{3.2}$$

where E_{form} is the formation energy of the unit cell (from the Materials Project), and $a_i * E_{\text{coh,i}}$ is the cohesive energy (taken from `matminer.utils.data.CohesiveEnergyData`) of the elemental species weighted by their molar fraction. The last term is the change of energy required to go from the elementary crystals to the individual atoms, while the first term is the energy required to go from the individual atoms to the final compound.

ElectronAffinity, see Ref. [55]. Computes a mean electroaffinity multiplied by the formal charge of the composing anions:

$$E_{\text{aff}} = a_i \cdot E_{\text{aff},i} \cdot Q_{\text{anion},i} \tag{3.3}$$

where $E_{\text{aff},i}$ is the affinity of anion i, a_i the molar fraction ($\sum_i a_i = 1$) and $Q_{\text{anion},i}$ the formal charge (oxidation state).

ElectronegativityDiff, see Ref. [55]. For each cation i the following quantity is computed:

$$D_{\text{cation},i} = \chi_{\text{cation},i} - M_{\text{ions}} \tag{3.4}$$

with M_{ions} is defined as

$$M_{\text{ions}} = \sum_i a_i \cdot \chi_{\text{anion},i} \tag{3.5}$$

where χ is the electronegativity and a_i the molar fraction of cation i. From the vector D_i, different statistics can be computed as final features, such as the mean and/or standard deviation.

IonProperty, see Ref. [30]. This class produces three features. The first one is a boolean set to TRUE if a neutral ionic compound is possible, i.e. if the sum of the weighted oxidation states by their molar fraction sum up to zero. The second and third feature computes the maximum and average ionic character of the composing atomic bonds (one for each pair of atoms, only based on composition and thus neglecting neighbors). This ionic character is based on the difference in electronegativity between the pair of atoms.

Miedema, see Ref. [60]. This feature represents the formation enthalpy of intermetallic compounds, solid solutions, and amorphous phases using the semi-empirical Miedema model. It is based on a set of self-consistent equations (see Ref. [60]), solved numerically, from the knowledge of the elemental properties. The final feature is the formation enthalpy of the binary/ternary alloy.

OxidationStates, see Ref. [55]. This featurizer simply gathers the oxidation states of the constituent elements and computes various statistics about them such as mean, maximum, minimum and variance.

TMetalFraction, see Ref. [55]. This featurizer generates a single feature obtained by computing the molar fraction of all transition metals (Ti, Fe, Co, Ni, Cu, Nb,...) in the compound's composition. For example $CaNiGe_2$ is transformed into 0.25.

ValenceOrbital, see Ref. [30]. For each element of the composition the valence electrons are determined as well as their orbitals. Then a variety of statistics about these valence electrons are computed. Some examples are the mean number of s-electrons, fractions

of p-electrons or the average deviation of the number of d-electrons of the constituent atoms.

YangSolidSolution, see Ref. [61]. Yang *et al.* developed two features that are useful for predicting whether metal alloys will form metallic glasses, crystalline solid solutions, or intermetallics. The idea is that the formation of compounds is both ruled by thermodynamics, and the sizes of the atoms and their respective mismatch. Therefore, the first feature, Ω, is related to mixing thermochimestry, by computing the balance between the mixing entropy and mixing enthalpy of the liquid phase:

$$\Omega = \frac{T_m \Delta S_{mix}}{\|\Delta H_{mix}\|} \tag{3.6}$$

Where T_m is the average melting temperature, ΔS_{mix} is the ideal mixing entropy, and ΔH_{mix} is the average mixing enthalpy taken over all pairs of elements in the considered alloy.

The second feature, δ, is related to the atomic size mismatch between the different elements in the material. It is defined as:

$$\delta = \sqrt{\sum_{i=1}^{n} c_i \left(1 - \frac{r_i}{\bar{r}}\right)^2} \tag{3.7}$$

where c_i and r_i are respectively the fraction and radius of element i, and $\bar{r}$ is the fraction-weighted average of the radii. This number deviates from zero all the more the radii are different. Atomic radii are taken from Ref. [62].

3.3.2 Structure

GlobalSymmetryFeatures, see Ref. [63]. Three features are generated related to the crystal symmetry. The first feature is the spacegroup number of the crystal. The second is the crystal system (cubic, tetragonal, orthorhombic, hexagonal, trigonal, triclinic or monoclinic) where the seven possible systems are mapped to an integer (1 to 7). The last feature is a boolean representing whether the structure is centrosymmetric, i.e. whether it has an inversion symmetry.

MaximumPackingEfficiency, see Ref. [47]. This single feature represents the packing efficiency of the crystal by measuring the volume fraction of occupied atoms. The algorithm first determines the largest radius for each atom, which equals the distance from the center of the atom to the closest Voronoi face. Then, the packing efficiency is computed as the following volume ratio:

$$\frac{1}{V_{cell}} \sum_i \frac{4}{3} \pi r_i^3 \tag{3.8}$$

where V_{cell} is the volume of the unit cell and r_i the largest radius (as defined before) of each atom i in the unit cell.

RadialDistributionFunction. The radial distribution function (RDF) or pair correlation function $g(r)$ represents the density of atoms at a certain distance r from a reference atom, see figure 3.3.2. More precisely, the number of atoms between two spheres of radius r and $r + dr$ centered around a reference atom is given by:

$$dN(r) = 4\pi r^2 g(r) dr. \tag{3.9}$$

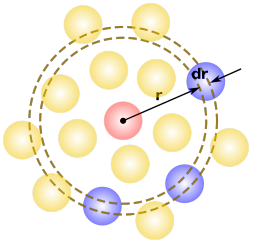

Figure 3.3.2: Radial distribution function $g(r)$ represents the probability to find a particle between a distance r and $r + dr$, illustrated by the dotted circles. Image taken from Ref. [64].

It can thus be seen as the probability of finding a particle at a distance r away from a given reference particle, relative to that for an ideal gas. Moreover, it can be used to calculate the coordination number, crystallinity and other quantities.

In practice, the algorithm computing the RDF gathers the distances between all atom pairs in the unit cell and forms a corresponding histogram with a certain bin size. This corresponds to a mean RDF over all sites of the unit cell as reference atom. Finally, this histogram is represented as a vector, where each element is a bin, giving the final feature.

DensityFeatures. Generates three density related features. The first is the density of the material (mass per unit volume). The second feature is the volume per atom or 'vpa' computed by dividing the unit cell volume by the number of sites present. The third and last feature is the packing fraction computed by dividing the sum of the atomic volumes (estimated by $\frac{4}{3}\pi r_i^3$, with r_i the species radius) by the unit cell volume.

BondFraction, see Ref. [65]. From the compound structure, all atomic bonds are identified by a nearest neighbor search. A bond is defined here only by the pair of atoms participating in this bond, for example Si-O. When all the bonds are enumerated, the fraction of each bond is returned as a feature element. For example for a structure containing 2 Li-O bonds and 3 Li-P bonds, this is represented by (Li-0: 0.4, Li-P: 0.6). It should be noted that for a complete dataset, the length of the feature vector (number of features) corresponds to the number of all possible occuring bonds, which is often in the order of a few thousands. Finally, this featurizer class requires for construction an instance of the pymatgen nearest neighbor class. In the present work, `CrystalNN()` is used.

CoulombMatrix, see Ref. [22]. The Coulomb-matrix, as it name suggests, represents the electrostatic Coulomb interactions within a structure or molecule in form of a matrix. It was originally proposed by Rupp *et al.* [22] for predicting atomization energies of a diverse set of organic molecules. The Coulomb matrix M off-diagonal elements are defined as,

$$M_{ij,i\neq j} = \frac{Z_i \cdot Z_j}{\|R_i - R_j\|} \tag{3.10}$$

and the diagonal elements are defined as,

$$M_{ii} = 0.5\ Z_i^{2.4} \tag{3.11}$$

where Z_i and R_i denote the nuclear charge and the position of atom i, respectively. Off-diagonal elements (M_{ij}) represents the Coulomb repulsion between two atoms i and j, while diagonal elements (M_{ii}) encode a polynomial fit of the atomic energies to nuclear charge of

atom i. It can be very simply extended to crystals by computing the Coulomb matrix for the *unit cell* (instead of a molecule). In order the use the Coulomb matrix for ML purposes, it should be transformed to a fixed length vector. Therefore, the eigenvalues of the Coulomb matrix are computed and ordered from largest to smallest. This flattens the matrix but does still not make it size invariant, because not all unit cells have the same number of atoms and therefore not the same number of eigenvalues, which equals the dimensionality of the Coulomb matrix. To solve this, the right end of the vector is zero-padded, which corresponds to extending missing eigenvectors to zero, such that every compound has the same length vector (this length is thus equal to the maximum number of species present in a particular compound).

Note that this feature has the following interesting properties [22]: (i) any system is uniquely encoded because stoichiometry as well as atomic configuration are explicitly accounted for, (ii) symmetrically equivalent atoms contribute equally, (iii) the diagonalized matrix M is invariant with respect to atomic permutations, translations, and rotations, and (iv) the eigenvectors are continuous with respect to small variations in inter-atomic distances or nuclear charges.

SineCoulombMatrix, see Ref. [66]. As described before, the Coulomb Matrix was initially designed for organic molecules. As it was shown to be successful, it was desirable to extend it to periodic crystals. Faber *et al.* [66] proposed the sine-Coulomb matrix, among others, in order to account for the interaction between repeating atoms in the crystal lattice. These long range electrostatic interactions over the crystal lattice can be represented by correcting the Coulomb matrix with an arbitrarily chosen two point potential $\Phi(\vec{r_i}, \vec{r_j})$:

$$M_{ij} = \begin{cases} Z_i \cdot Z_j \cdot \Phi(\vec{r_i}, \vec{r_j}) \\ 0.5 \cdot Z_i^{2.4} \end{cases} \tag{3.12}$$

This two-point potential is a function of the position of two atoms A and B, representing their interactions over the infinitely repeating grid. This potential should (i) be periodic with respect to the crystal lattice, (ii) have the same value for two equivalent atoms in neighboring cells and (iii) tend to infinity when these two atoms take the same position. A possible solution is found by setting

$$\Phi(\vec{r_i}, \vec{r_j}) = \|\mathbf{B} \cdot \sum_{k=\{x,y,z\}} \hat{e_k} sin^2[\pi \hat{e_k} \mathbf{B^{-1}} \cdot (\vec{r_i} - \vec{r_j})]\|_2^{-1} \tag{3.13}$$

where $\hat{e_x}, \hat{e_y}, \hat{e_z}$ are the coordinate unit vectors and $\mathbf{B}$ the matrix formed by taking for each line the basis vectors of the lattice.

Equations (3.12) and (3.13) defines the sine-Coulomb matrix representation of a periodic crystal structure. It has the benefit of depending only on the relative positions of the atoms in the unit cell and hence, has a minimal computational load.

ChemicalOrdering, see Ref. [47]. This featurizer generates a single feature describing the chemical order of the species within the structure, by computing how much the structure differs from random. The first step of this calculation is to determine the nearest neighbor shell of each site s. Then, a degree of order $\alpha(t, s)$ is determined, for each site s and

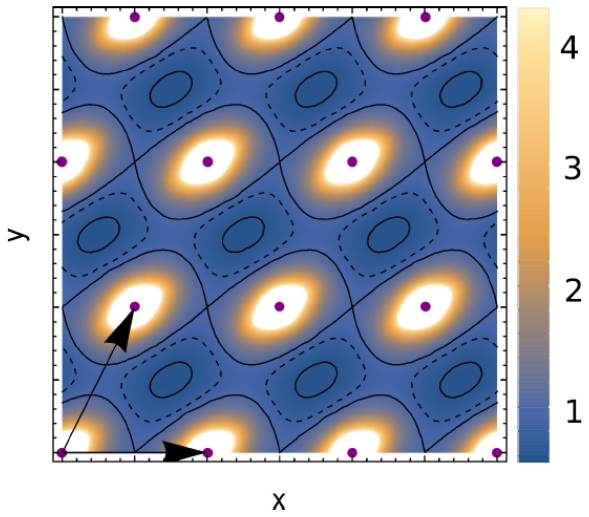

Figure 3.3.3: Illustration of $\Phi(\vec{r_1}, \vec{r_2})$ used in the sine-Coulomb matrix representation, Eq. (3.13), for a 2D crystal lattice represented by the dark arrows and purple dots. One atom is moved following $r_1 = (x, y)$ while another is fixed at the origin $r_2 = 0$. Image taken from Ref. [66].

chemical species t defined as:

$$\alpha(t, s) = 1 - \frac{\sum_n w_n \delta(t - t_n)}{x_t \sum_n w_n} \tag{3.14}$$

where w_n is the weight associated with a certain neighbor (i.e. how far it is from the central site), t_n is the type (chemical element) of the neighbor, and x_t is the molar fraction of type t in the structure. For atoms that are randomly dispersed in a structure or unary compounds, this formula yields 0 for all types. For structures where each site is surrounded only by atoms of another type, this formula yields large values of α. The mean absolute value of this parameter across all sites is used as a feature.

EwaldEnergy, see Ref. [63]. This featurizer generates a single feature representing the Ewald energy based on Coulombic interactions using the charges defined in the structure. For this, the corresponding `EwaldSummation` function from *pymatgen* is used.

StructuralHeterogeneity, see Ref. [47]. This featurizer computes several statistics derived from the Voronoi tessellation of a structure. First, the mean bond length of each site is computed. This is done by taking all neigbours (i.e. atoms that share a common Voronoi face) of a site and taking the corresponding mean. From this, three features are generated: the minimum average site bond length, the maximum average site bond length and the mean absolute deviation of the average site bond length (over the sites). They are then normalized by dividing by the mean (over the sites) average (over the neighbors) site bond length. Finally, a last feature is formed by computing the mean absolute deviation of the Voronoi cell volume (over the sites) divided by the mean Voronoi cell volume.

XRDPowderPattern, see Ref. [66]. For each crystal a powder diffraction pattern can be obtained that forms a fingerprint of the crystal. The `XRDPowderPattern` featurizer represents this diffraction pattern as obtained from pymatgen where each feature $f_{2\theta}$ represents the diffraction intensity at angle 2θ. The intensities are smeared and normalized

according to a Gaussian kernel, i.e. the intensities are seen as a probability density function (PDF) of a random variable (the angle here) and are estimated by the kernel density estimation in a non-parametric way.

3.3.3 Site

Site based features are features based on the individual sites of the unit cell, i.e. a specific atomic position in the unit cell. Some examples are the coordination number of a site (number of neighbors) or the mean bond length of a specific site. More generally, one can introduce the concept of Order Parameter (OP). An OP is a mathematical construct that provides a numerical measure of the local environment of an atom. One of the simplest OP is the coordination number (CN) q_{CN} [67]:

$$q_{\mathrm{CN}} = \sum_{j \neq i} S\left(\|\mathbf{r}_j - \mathbf{r}_i\|\right)$$

where $\mathbf{r}_j$ is the position of neighbor atom j and S a function that is 1 if the argument is smaller then a threshold distance r_{cut} and $S = 0$ otherwise. This OP can be further sophisticated by using a distance-weighted function such as the Fermi function [67] :

$$q_{\mathrm{CN,\ Fermi}} = \frac{1}{\exp\left[\kappa\left(d_{i,j} - r_{\mathrm{cut},i}\right)\right] + 1}$$

where κ^{-1} represents the transition width from 1 towards 0 as $d_{i,j}$ increases. Note that these OPs do not encode any geometrical arrangement of the atoms. Therefore more advanced OP have been introduced, and will be discussed below.

Finally, because these type of features are intrinsically tailored for a specific site, they are more suited for applications involving individual site characteristics, such as vacancy energies or diffusion coefficients. The different featurizers within matminer are detailed below.

OPSiteFingerprint, see Ref. [68]. In order to overcome the limitations of simpler OPs, Zimmerman *et al.* [68] introduced a set of OPs that provide information on how closely an environment resembles a given structural motif. Different common structural motifs are shown in figure 3.3.4.

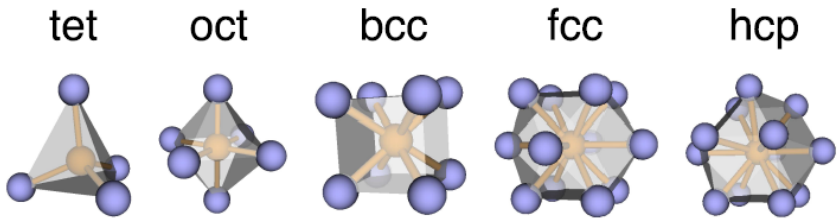

Figure 3.3.4: Basic structural motifs that frequently recur in various material databases, from left to right: tetrahedron (tet), octahedron (oct) as well as body-centered cubic (bcc), face-centered cubic (fcc), and hexagonal close packed (hcp) motifs. The central atom and bonds are shown in orange, whereas the coordinating atoms are displayed in blue. From Ref. [68].

For each structural motif a corresponding OP is formed that describes how closely it matches the motif, from 0 (no match) to 1 (match). The computation is done in two steps. First, the first neighbor shell is localized. This is done by determining the distance $d_{min,i}$ of the nearest neighbor, of a given site i and subsequently considering all additional sites that are at maximum $r_{\mathrm{cut},i} = (1+\delta)d_{\mathrm{min},i}$ apart from site i where δ denotes a neighbor-finding tolerance. Two other approaches of finding the neighbors are presented in the reference article.

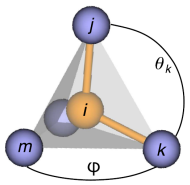

Figure 3.3.5: Definition of atom indices, i, j, k and m as well as angles, θ (polar) and ϕ (azimuth). Used for the OPs introduced by Zimmerman *et al.* From Ref. [68].

The second step, once the first neighbor shell is determined, is to give a matching score to a specific motif. This is achieved by setting up a spherical coordinate system around the central atom i and two neighbors j and k that forms a plane from where the polar angles θ (within plane) and ϕ (out of plane) are defined, see figure 3.3.5. This enables to localize other neighbors $l, m, ...$ If a neighbor is not located at angles that are commensurate with the expected positions of the underlying motif, the decline in reward for being away from the perfect position follows a Gaussian function. Conversely, a full reward is given if any expected remaining position is exactly assumed. This procedure gives rotationally invariant OPs, because all possible pair of neighbors for defining the coordinate system are taken and averaged.

As an example, the tetragonal OP is given below:

$$q_{\mathrm{tet}} = \frac{1}{N_{\mathrm{ngh}}\left(N_{\mathrm{ngh}}-1\right)\left(N_{\mathrm{ngh}}-2\right)} \sum_{j\neq k}^{N_{\mathrm{ngh}}} \left\{ \exp\left[\frac{-\left(\theta_k - 109.47°\right)^2}{2\Delta\theta^2}\right] \sum_{m\neq j}^{N_{\mathrm{ngh}}} \cos^2(1.5\varphi) \exp\left[\frac{-\left(\theta_m - 109.47°\right)^2}{2\Delta\theta^2}\right] \right\}$$

where the angles θ and ϕ are defined as above, N_{ngh} denotes the number of neighbors bonded to the central atom i in a motif and $\Delta\theta = 12°$ is the Gaussian width. It basically computes the deviation from the expected angles (109.47° polar and 120° azimuth) in a Gaussian fashion.

Finally, the `OPSiteFingerprint` featurizer computes a bunch of OPs corresponding to various structural motifs, which forms the outputted feature vector.

VoronoiFingerprint, see Ref. [69]. This featurizer computes various statistics about the Voronoi tesselation of the compound structure. In mathematics, Voronoi tesselations are geometrical constructs that partition the space into regions based on the distance to points in a specific subset of the space [70]. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells. The Voronoi diagram of a set of points is dual to its Delaunay triangulation. It is named after Georgy Voronoi, and is also called a Voronoi tessellation, a Voronoi decomposition, a

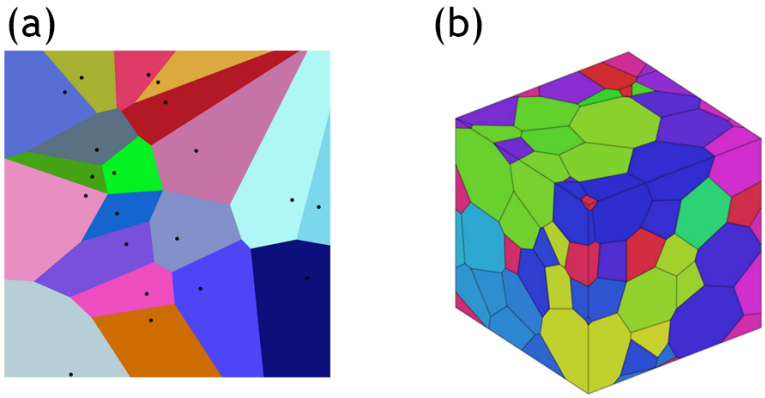

Figure 3.3.6: Illustrative example of two Voronoi tesselations in 2D (a) and 3D (b), where the seeds are represented as black dots. Images taken from Refs. [70] (a) and [71] (b).

Voronoi partition, or a Dirichlet tessellation. Two examples are given in figure 3.3.6 (2D and 3D).

Once the Voronoi tesselation determined, various statistics about the site Voronoi cell (or polyhedra) is computed. The first is a vector N where N_i, called the Voronoi indices, represents the number of i-edged facets around the considered site, where i goes from 3 to 10. For example, for a bcc lattice $N = [0, 6, 0, 8, ...]$ meaning that there are six 4-facets and eight 6-facets. For fcc/hcp lattice $N = [0, 12, 0, 0, ...]$ and for icosahedra $N = [0, 0, 12, 0, ...]$.

The second set of statistics are i-fold symmetry indices computed as:

$$S_i = \frac{N_i}{\sum_j N_j} \tag{3.15}$$

which are basically normalized Voronoi indices such that $\sum_j S_j = 1$. Optionally these indices can be weighted by their relative distance to the central site, see the matminer API for more information.

Other statistics are the Voronoi-polyhedron volume and surface, Voronoi-volume statistics of subpolyhedra formed by each facet and the center site, the facets area and finally nearest-neighbors distances (defined as the distance to the corresponding facet).

Combining these statistics into a single vector forms the feature of this featurizer.

CrystalNNFingerPrint. This featurizer first computes a score for various CN (0 to 10 for example), giving how well the local environment matches that CN. This score is computed using the `CrystalNN` class from pymatgen, which contains the `cn_weights` attribute giving a weight (or score) for a CN based on the neighbors distances. This gives a first set of features. Then, various other OPs are computed for that site that are more related to the structural motifs via the pymatgen `LocalStructOrderParams` class. This class is mainly a container to the previously explained OPs from Ref. [68], but other motifs are considered too such as the body-centered cubic OP from Ref. [72] or pentagonal bipyramids from Ref. [73]. Once those OPs are computed they are multiplied by their corresponding coordination score to form the final feature vector.

GaussianSymmFunc, see Refs. [14] and [74]. This featurizer will encode the local

environment of a site into various symmetry functions instead of Cartesian coordinates. It was originally proposed by Behler *et al.* [14] to fit potential energy surfaces for molecular dynamics. The general idea is to form a set of descriptors that encode the local environment of an atom (such as neighbor distances and angles) that is invariant with respect to rotations of the structure and has a fixed number of dimensions. This latter is due to the fact that most ML algorithms have a fixed input dimension. Moreover these descriptors should be continuous with respect to the atomic positions, in order to guarantee derivability, a necessary condition to compute force fields. The proposed solution is to use many-body symmetry functions that simultaneously depend on the positions of all atoms inside a certain cutoff sphere. The proposed set of radial symmetry functions are:

$$G_i^1 = \sum_j f_{\mathrm{c}}\left(R_{ij}\right) \tag{3.16}$$

$$G_i^2 = \sum_j e^{-\eta\left(R_{ij}-R_s\right)^2} \cdot f_{\mathrm{c}}\left(R_{ij}\right) \tag{3.17}$$

$$G_i^3 = \sum_j \cos\left(\kappa R_{ij}\right) \cdot f_{\mathrm{c}}\left(R_{ij}\right) \tag{3.18}$$

where R_{ij} is the distance between the site i and every neighbor j, R_s a suitable shift distance, η and κ are parameters ruling respectively the width of the Gaussian and the periodicity of the cosinus and finally f_c is a cutoff function:

$$f(r) = \begin{cases} 0.5\left[cos(\frac{\pi r_{ij}}{R_c}) + 1\right] & \text{if} \quad r_{ij} \leq R_c \\ 0 & \text{if} \quad r_{ij} > R_c \end{cases} \tag{3.19}$$

with R_c is the cutoff radius. Equations (3.16)-(3.18) can be seen as the integral of the product of the RDF with a unit, Gaussian and cosine function respectively (itself multiplied by the cutoff function). In particular equation (3.16) is the sum of all neighbors weighted by the cutoff function and equation (3.18) can be seen as a Fourier decomposition. A graphical representation of the radial symmetry functions can be found in figure 3.3.7.

By computing $G^1, G^2, G3$ at various values for η and κ a feature vector encoding the radial information can be obtained.

Similarly a set of angular function is proposed:

$$\begin{aligned} G_i^4 = 2^{1-\zeta} \textstyle\sum_{j,k\neq i}^{\text{all}} \left(1+\lambda\cos\theta_{ijk}\right)^{\zeta} \cdot e^{-\eta\left(R_{ij}^2+R_{ik}^2+R_{jk}^2\right)} \\ \cdot f_{\mathrm{c}}\left(R_{ij}\right) \cdot f_{\mathrm{c}}\left(R_{ik}\right) \cdot f_{\mathrm{c}}\left(R_{jk}\right) \end{aligned} \tag{3.20}$$

$$\begin{aligned} G_i^5 = & 2^{1-\zeta} \sum_{j,k\neq i}^{\text{all}} \left(1+\lambda\cos\theta_{ijk}\right)^{\zeta} \cdot e^{-\eta\left(R_{ij}^2+R_{iz}^2\right)} \\ & \cdot f_{\mathrm{c}}\left(R_{ij}\right) \cdot f_{\mathrm{c}}\left(R_{ik}\right) \end{aligned} \tag{3.21}$$

where θ_{ijk} is the angle formed by atoms i,j and k. The parameter λ is taken once as +1 and once as -1 (shifting the maxima of the cosine function). Parameters ζ and η are again

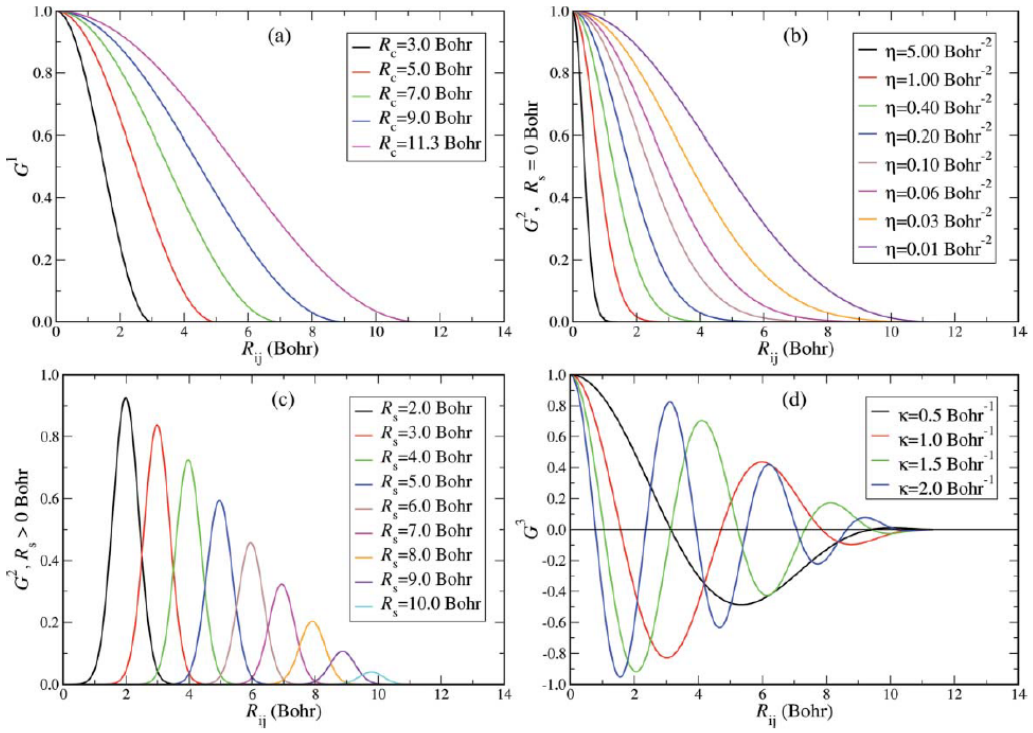

Figure 3.3.7: Radial symmetry functions G^1, G^2, and G^3 for an atom with one neighbor only as used in the `GaussianSymmFunc` featurizer. For the G^2 and G^3 functions a cutoff $R_c = 11.3$ *Bohr* has been used, the Gaussian width of the shifted G^2 functions in panel (c) is $\eta = 3.0$ $Bohr^2$. From Ref. [14].

taken over a varying range in order to obtain a set of features representing the angular geometry.

Commonly, around 50 features are generated by combining 50 different symmetry functions (G^1 to G^2 and their parameters) in order to represent a single environment.

ChemEnv, see Ref. [75]. This featurizer generates a set of OPs, the features, representing the closeness of the local environment to various perfect coordination environment (tetrehedral, trigonal prism,...) in form of a percentage. It is therefore very similar to `OPSiteFingerprint`, but the actual implementation is different.

It first determines a set of neighbors of the central site through a Voronoi approach similar to what was proposed by O'Keeffe [76]. Atoms that share a common Voronoi face with the central atom are considered to be neighbors of the central site. Then, two cutoff parameters are introduced to discard some neighbors. The first cutoff parameter α discards neighbors that are localized further than αd_{min} where d_min is the distance to the closest neighbor. The second cutoff parameter γ discard atoms that have a solid angle formed by the common Voronoi face that is smaller than $\gamma\Omega^{max}$ where Ω^{max} is the largest solid angle formed by a common Voronoi face. Additionally other neighbors can be discarded based on their oxidation states. For example in ionic solids, cations neighboring other cations should be discarded.

Once a set of neighbors is established, it can be compared to various model polyhedra. For this, the Continuous Symmetry Measure (CSM) introduced by Pinsky and Avnir [77] is used. It defines a distance S between a distorted structure P and a perfect G-symmetric

structure Q, where G is a specific symmetry group:

$$S = \min \frac{\sum_{k=1}^{N} \|Q_k - P_k\|^2}{\sum_{k=1}^{N} \|Q_k - Q_0\|^2} \times 100 \tag{3.22}$$

where $\{Q_k, k = 1, 2, \ldots, N\}$ are the coordinates of the N vertices of Q, $\left\{\hat{P}_k, k = 1, 2, \ldots, N\right\}$ the coordinates of the N vertices of P and Q_0 is the coordinate vector of the center of mass of the investigated structure:

$$Q_0 = \frac{1}{N} \sum_{k=1}^{N} Q_k \tag{3.23}$$

In order to have the best match with the perfect structure P, this latter should be rotated, dilated and translated, in order to minimize the distance S, which is represented with the *min* keyword. These operations can mathematically be written as:

$$P_k = A\mathbf{R}P_{0k} + \mathbf{T} \tag{3.24}$$

and the corresponding minimization problem:

$$J = \sum_{k=1}^{N} \|Q_k - P_k\|^2 = \sum_{k=1}^{N} \|Q_k - (A\mathbf{R}P_{0k} + \mathbf{T})\|^2 \tag{3.25}$$

with $\mathbf{R}$ a 3x3 rotation matrix, $\mathbf{T}$ a 3x3 translation matrix and A an isotropic scaling factor. This minimization is central to the CSM algorithm and is discussed in more detail in the original paper [77].

Once the optimal set of coordinates are found for P, the corresponding distance S can be computed. It should be noted that S is bounded within the interval $[0, 100]$. The symmetry measure attains its minimum value ($S = 0$) when the structure has exactly the investigated symmetry.

Going back to the `ChemEnv` featurizer [75], it uses a specific *strategy* to find the different percentages for the different coordination environments based on the CSM. First, for more robustness, different sets of neighbors are considered by varying the cutoff parameters: $\alpha \in [1.15, 2.0]$ and $\gamma \in [0.05, 0.75]$. Secondly, for each set of neighbors (corresponding to a set of cutoff parameters) different fractions or percentages are given for different possible polehedra. This is done by multiplying two factors. The first is a weight that is a decreasing monotonic mapping of the CSM distance, giving higher weights to closer structures. The second weight factor favor neighbor-sets of higher coordination (even if they are more distorted, i.e. higher CSM distance).

Finally, this framework enables one to generate a feature vector giving a percentage to the closeness of different local coordinations, and doing so, it considers local environments as a superposition of different polyhedra rather then one alone, giving a more robust descriptor.

AGNIFingerprints, see Ref. [78]. This feature represents the local environment of an atom by computing the integral of the product of the radial distribution function and a

Gaussian window function. It is therefore very similar to the work of Behler *et al.* [14]. It has the advantage of giving a vector $G(\eta)$ that is not sparse compared to the RDF, and thus much more suited for similarity comparison that are computed by taking the L^2-norm of the difference or the scalar product between two vectors, which would respectively be one or zero for the RDF. Originally, Botu *et al.* [78] used it to fit empirical energy potentials. This OP is defined as:

$$G_i(\eta) = \int R_i(r) e^{-(\frac{r_{ij}}{\eta})^2} dr \tag{3.26}$$

where i is the considered site, $R(r)$ the RDF and r_{ij} the distance between site i and neighbor j. By introducing the same cutoff-function $f(r_{ij})$ as Behler *et al.* [14] and discretizing the integral, one has:

$$G_i(\eta) = \sum_{i \neq j} e^{-(\frac{r_{ij}}{\eta})^2} f(r_{ij}) \tag{3.27}$$

which forms the undirected AGNI fingerprint, a vector containing $G(\eta)$ at different η's, a first set of features. The cutoff-function $f(r_{ij})$ is taken as:

$$f(r) = \begin{cases} 0.5\Big[cos(\frac{\pi r_{ij}}{R_c}) + 1\Big] & \text{if} \quad r_{ij} \leq R_c \\ 0 & \text{if} \quad r_{ij} > R_c \end{cases} \tag{3.28}$$

In order to introduce directionality to the feature, which is interesting for fitting vector fields (e.g force fields), the previous equation is extended to direction-resolved atomic fingerprints $V_i(\eta) = \{V_i^x(\eta), V_i^y(\eta), V_i^z(\eta)\}$ as follows:

$$V_i^k(\eta) = \sum_{i \neq j} \frac{r_{ij}^k}{r_{ij}} e^{-(\frac{r_{ij}}{\eta})^2} f(r_{ij}) \quad , \quad k \in \{x, y, z\} \tag{3.29}$$

where r_{ij}^k is the k-th component of $(\vec{r_i} - \vec{r_j})$. This gives a second set of features $V_i(\eta)$. It should however be noted that this latter set of features depend on the axes of the reference system and is therefore not invariant with respect to rotations.

CoordinationNumber, see Refs. [79] and [80]. This featurizer generates a unique feature representing the coordination number of a given site, i.e. the number of nearest neighbors. In order to determine the nearest neighbors, it uses one of pymatgen `NearNeighbor` classes such as `VoronoiNN`, `JmolNN`, `MinimumDistanceNN`, `MinimumOKeeffeNN` or `MinimumVIRENN`. In the present work, `VoronoiNN` is used. Neighbors are determined based on the surface of the common Voronoi face with the central site. This surface can be translated into a weight for each neighbor and can optionally be incorporated into the coordination number by summing over the weights of the different neighbors. This weighted sum avoids that the coordination number changes abruptly with small perturbation of the local environment.

GeneralizedRDF, see Ref. [81]. This featurizer generalizes the RDF by using non-rectangular bins. Remember that for the RDF, each feature represents the number of bonds

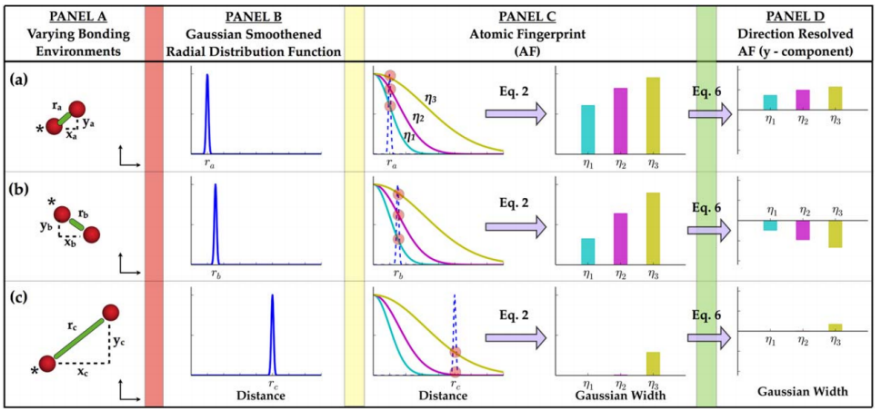

Figure 3.3.8: Schematic representation of the AGNI features. They are formed by transformation the RDF using Gaussian functions on an eta-grid as indicated by the colored lines in panel C. The direction resolved fingerprints are showed in panel D. Image taken from Ref. [78].

that are within a certain length interval, which can be written as:

$$x_k = \int_{k\Delta}^{(k+1)\Delta} g(r)dr \tag{3.30}$$

where g(r) is the RDF, Δ the bin width and x_k the k-th feature ($k \in [0, \lceil r_{cuttof}/\Delta \rceil]$, with $\lceil \cdot \rceil$ the ceiling function).

The GRDF generalizes this latter to:

$$x_k = \int_{0}^{r_{cuttof}} H(r - ((k + \frac{1}{2})\Delta)g(r)dr \tag{3.31}$$

where $H(r)$ is a binning function centered on the considered interval. The bins are now potentially overlapping and thus not mutually exclusive. In the present work the default Gaussian preset is used for $H(r)$. Note that taking $H(r) = 1$ from $k\Delta$ to $(k+1)\Delta$ and 0 otherwise is equivalent to equation 3.30. The RDF is thus a particular case of the GRDF. Note that x_k can also be seen as:

$$\begin{cases} f(r) = \sum_j H(r_{ij} - r) \\ x_k = f((k + \frac{1}{2})\Delta) \quad , \quad k \in \mathbb{N} \end{cases} \tag{3.32}$$

where j are the neighbors of site i. In other words, the RDF is convoluted with H(r) (if one assumes that the RDF are Dirac-δ functions centered around the neighbor positions).

BondOrientationalParameter, see Ref. [73], [81]. In order to asses local symmetries to local bonds around a particular site, these bonds are associated to spherical harmonics. Spherical Harmonics are a set of radial functions Y_{lm} that are solution to the Laplace equation,

$$Y_{lm}(\theta, \phi), \tag{3.33}$$

where l and m are integer parameters with $m \in [-l, l]$. Spherical harmonics are therefore solutions of the angular part of the Schrödinger equation for a radial potential (e.g hydrogen atom), where l is related to the angular momentum and m the magnetic momentum. A graphical representation of the first spherical harmonics are drawn in figure 3.3.9.

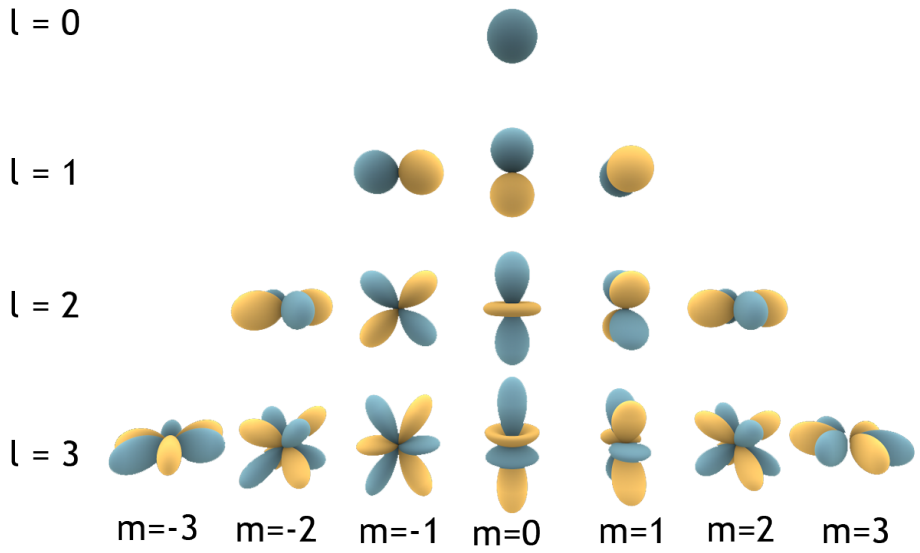

Figure 3.3.9: Illustration of the first few real spherical harmonics $Y_{lm}(\theta, \phi)$.Blue zones (resp. yellow zones) represent regions where the function is positive (resp. negative). The distance of the surface from the origin indicates the value of $Y_{lm}(\theta, \phi)$ in the angular direction (θ, ϕ). Figure adapted from Ref. [82].

As spherical harmonics represent quite well different types of angular symmetries, Steinhardt *et al.* proposed to use them to describe the local environment of an atomic site. The idea is to associate as set of numbers Q_{lm} to a particular site, representing how well it matches with a particular spherical harmonic [73],

$$Q_{lm}(\vec{\mathbf{r}}) \equiv Y_{lm}(\theta(\vec{\mathbf{r}}), \phi(\vec{\mathbf{r}})) \tag{3.34}$$

where $\theta(\vec{r})$ and $\phi(\vec{r})$ are the polar angles of a particular bond with midpoint $\vec{r}$. In order to take into account all bonds of a site, a mean $\overline{Q}_{lm}$ is taken over all bonds:

$$\overline{Q}_{lm} \equiv \langle Q_{lm}(\vec{r}) \rangle \,. \tag{3.35}$$

Moreover, to guarantee rotational invariance, the following combination is defined:

$$Q_l \equiv \left(\frac{4\pi}{2l+1} \sum_{m=-l}^{l} \left\| \overline{Q}_{lm} \right\|^2 \right)^{1/2} \tag{3.36}$$

which forms the final feature vector by taking l from 1 to 10.

This feature vector forms a fingerprint of the local symmetry, for example for a cubic symmetric environment, the first nonzero element is Q_4, i.e. when $l = 4$. For an icosahedrally oriented system the first nonzero element occur for $l = 6$.

The Bond Orientational Parameter was initially introduced by Steinhardt *et al.* to study order in liquids and glasses in function of the temperature. It has later been used by Seko *et al.* [36] in the machine learning field to predict melting temperatures of single and binary compounds.

LocalPropertyDifference, see Ref. [47]. This featurizer computes the difference in elemental properties between the site and its neighbors, such as the mean difference in atomic number. It is therefore necessary to first determine the neighbors, which is done here by the Voronoi-tesselation approach. Each neighbor is weighted proportionally to the Voronoi facet (face shared between the site's cell and the neighbor's cell). The local property difference is then computed by the following weighted average:

$$p_{diff} \frac{\sum_n A_n \|p_n - p_0\|}{\sum_n A_n} \tag{3.37}$$

where A_n is the weight of neighbor n, p_n the property corresponding to neighbor p and p_0 the property corresponding to the central site. This finally forms the feature vector by computing p_{diff} over different elemental properties.

AverageBondLength, see Ref. [83]. This class generates the mean bond length over all Voronoi neighbors, by means of a weighted average, similar to equation (3.37) where p_n is replaced by the corresponding bond length.

AverageBondAngle, see Ref. [83]. The mean bond angle is computed between the site and the set of pairs of closest neighbors. More precisely, a matrix A is first generated containing the bond angle A_{ij} between site s, neigbor i and neigbor j, for all neigbors determined with a pymatgen `NearNeigbor` class such as `VoronoiNN`. Then for each line (i.e. neighbor) only one angle is kept, the smallest, called the neighbor angle. Finally the average is taken over all neighbor angles. Note that the angles are not weighted here.

3.4 List of features

All classes presented above generate together a total of 210 composition-related features, 517 structure-related features and 523 site-related features. They are listed in appendices D.1, D.2 and D.3 respectively. As can be seen from these appendices, each feature has a corresponding coded label, **C**xyz for **C**omposition, **S**xyz for **S**tructure, and **L**xyz for site (**L**ocal environment) features, with xyz a 3-digit integer identifier. These coded labels will be used in subsequent graphs to make visualization easier.

3.5 Conclusion

This chapter mainly introduced many of the available featurizers, in order to transform a compound into a fixed length vector capturing most of its defining properties. These features (or descriptors) were divided in three subgroups: composition, structure and site as provided by *matminer*. They were explained in details, in order to enhance understanding of the upcoming work. In particular, it was seen how they vary broadly in their nature, complexity and dimensionality. One could arguably ask how these different features are related to each other (or with targets), in order to find some redundancy or patterns between properties. This will be the task of the next chapter.

Chapter 4

Feature selection

4.1 Introduction

From the different featurizer classes presented in the previous chapter, it was seen that the set of possible discriptors is really enormous. If one would use all of them, the feature vector would easily reach a few thousand of entries, if not more when considering mathematical functions of those (e.g logarithm-, square-, etc. functions), which is often not desirable. One could think that more features means more information for the regression algorithm, as it can choose which one are the most useful during the training phase and put zero weight to meaningless features. However this is not true for most regression algorithms and having huge input vectors is often not desirable for many reasons. First, most regressors have a time complexity that scales with the number of features, meaning that having too many features would slow down the learning and prediction process and thus make it not a viable solution anymore [84].

Second, most regressors (including neural networks, see Ref. [85]) suffer from what is known as the curse of dimensionality, as explained in Ref. [86]. Easily explained, by increasing the number of features, the dimensionality of the input space increases which causes overfitting, due to an exponential increase in void space. More information about this will follow further in this chapter.

It is therefore crucial to limit the input space by using only an acceptable number of features. Several methods have been proposed to reduce the dimensionality of ML-problems. They can mainly be divided in two categories: feature selection and feature extraction (each of these categories could optionally again be divided into their linear or non-linear nature). Feature selection chooses a bunch of features from the initial feature set without changing them by merely selecting 'useful' features with respect to some 'usefulness' criterion. Feature extraction on the other hand (or sometimes called feature projection) creates a bunch of new features from the complete existing set of features by projecting it on a particular subspace. Unfolding manifolds is an example of non-linear dimensionality reduction or feature extraction. Motivated by its physical interpretability, feature selection is first investigated in the present work.

The goal of this chapter is double. First, statistical dependency between features *and* between features and target will be studied, offering a more profound insight and understanding of the previously presented descriptors. Secondly, from these results, a selection algorithm

based on a relevance-redundancy criterion will be presented.

4.2 Methodology

When selecting features for a model, two concepts are of importance: relevance and redundancy. Relevance will indicate if a descriptor is relevant, i.e useful for the model in order to predict the target output. It is evident that for example a feature containing the date when a material has been added in the database is totally useless. The second concept, redundancy, is more subtle and mainly concerns the fact that two features that are in fact very similar are often not desirable. Again, as an extreme case, a feature that is a multiple of another one can be safely discarded.

How the features are actually chosen will be explained in a later section. The first task is to find how we will quantify relevance and redundancy. The relevance will be tackled by measuring some statistical score of dependency between a feature and the target output. Similarly, the redundancy will be tackled by analyzing some statistical score between two features. These dependencies or relations will additionally give us more insight on the features, as we will see which properties are closely related together or which properties are closely related to the target. Several choices are possible as *statistical score* of dependence, but the mutual information is chosen here for the reasons that will follow.

4.2.1 Mutual information and normalized mutual information

Given two random variables X and Y, we are interested in a score that measures how the knowledge of Y could reduce our uncertainty on the value of X. A well known statistical score is the Pearson correlation defined as:

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{4.1}$$

where $Cov(X,Y)$ is the covariance between X and Y, and σ the standard deviation. Unfortunately the Pearson correlation has some displeasing properties for our goal.

First, the correlation is a *linear* measure, which means that an absence in correlation does not mean absence of relation. As an examples $y = x^2$, has a zero correlation. Second, it is parametric (makes the hypothesis of a linear model) and is moreover very sensitive to outliers.

To go beyond these limitations, the more general concept of mutual information (MI) is used, which measures the decrease in *information entropy* when the second random variable is known. Formally, the information entropy H of a continuous random variable X is defined as:

$$H(X) = -\int P_X(x) log(P_X(x)) dx \tag{4.2}$$

where $P_X(x)$ is the probability density function (pdf) of X. From this the mutual information $\mathrm{MI}(X,Y)$ between two continuous random variables X and Y can be defined as,

$$\mathrm{MI}(X,Y) = H(X) - H(X|Y). \tag{4.3}$$

It is always non negative and smaller than $\min(H(X), H(Y))$. When two random variables are independent the MI is zero.

The difficulty resides in the estimation of the MI from a sample representing a random variable, which requires to estimate the pdf using non parametric methods. In the present work, a KNN-based method is used for entropy estimation, as explained in Ref. [87], and implemented by the scikit-learn library.

An additional difficulty arises from the fact that the MI is not upper bounded. We therefore introduce the normalized mutual information (NMI) defined as,

$$\mathrm{NMI}(X, Y) = \frac{\mathrm{MI}(X, Y)}{(H(X) + H(Y))/2} \tag{4.4}$$

which is bounded between 0 and 1. Note that both the MI and NMI are symmetric measures. Unfortunately, in practice due to its non-parametric estimation, the obtained results are often not perfectly symmetric, although being not far from it (which moreover causes the NMI to sometimes go beyond 1). To limit these impractalities, averages are taken over pairs of corresponding NMI's by taking the symmetric part of the cross-NMI matrix.

4.2.2 Datasets

Two datasets will be used for this study. First for the cross-NMI between features, materials having an energy of at most 25meV above the convex hull in the Materials Project [50] will be used which corresponds to approximately 40,000 samples. This is because for the cross-NMI's we do not need to have any particular target property, thus let us take the full advantage of a larger dataset. Concerning the feature-target NMI, we are of course limited by the number of materials that have been computed with the considered target. In the present work, we are interested in predicting vibrational-thermodynamic properties as computed in reference [11], a high-throughput density-functional perturbation theory phonon study for inorganic materials, which contains 1245 samples. The vibrational entropy per atom and per mol is chosen as target.

4.2.3 Workplan

The upcoming work is divided in two main parts by first computing the NMI between features mutually followed by computing the NMI between features and target. The first part, called the cross-feature analysis, will compute the NMI between all pairs of possible features from the composition, structure and site module, which can be represented in matrix form, see figure 4.2.1. Because this matrix is symmetric, only 6 submatrices (see shaded regions in the previous figure) will be computed and shown, which is moreover less imposing to see than the complete matrix at once.

4.2.4 Importance of convergence

It is important to note that we are dealing with a finite dataset that is a sample from a larger population, namely the material space (which is moreover probably unlimited). This means that it is possible to introduce a bias by wrongly sampling the material space. The problem is that we want to have very general results concerning the relevance and redundancy among the features, which requires a correct representation of the material

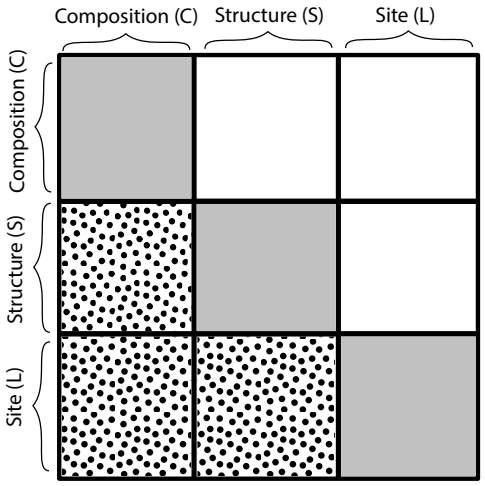

Figure 4.2.1: Schematic representation of the cross-feature NMI matrix composing the three feature modules: composition, structure and site. The shaded grey submatrices correspond to intra-module NMI's while the dotted submatrices represents inter-module NMI's. Only these 6 submatrices will be computed and shown.

space. Therefore, for each NMI study between features, random samples of increasing size are taken from the previously presented dataset. For each of these samples the NMI is computed, and from this the differences in NMI between samples of different sizes is calculated. When these differences tend to zero, we can assure convergence with respect to the sample size (and thus the population representation). [1]

These differences in NMI forms the basis of the convergence study, and will be shown graphically in heatmaps such as the one shown in figure 4.4.1. The remaining paragraphs of this subsection discuss why a non-zero difference is seen and how to interpret these differences.

Three main types of differences are seen: first, lightly blue colored areas corresponds to an increase in mutual information. They show that new relations in the probability density function are found thanks to the increase in samples, that better mimics the underlying relation, and are thus very important.

The second type is indicated by a light reddish color, thus negative differences. They show a decrease in mutual information, due to an increase in samples that are in fact more randomly distributed then before. Or more correctly, the smaller sample-set was following by chance some relation (a line for example) and this relation was later broken by sampling more. This explanation forms the *hypothesis* for this behavior and is further confirmed by two facts:

1. Negative differences are much rarer then positive differences

[1] It is in fact very difficult to achieve and verify convergence *over the hole material space*, as our dataset is a sample of the Materials Project which is in turn a sample of the material space and could both be biased on itself. But if this were true most things would be overly complicated, and the main purpose of this field, i.e. generalization would be fairly impossible. Fortunately, it is probably not entirely the case and assuring convergence on our dataset is certainty sufficient to draw interesting results. At least we know that we can *interpolate* our dataset if not more...

2. Negative differences between two datasets doubling in size are much less frequent when the sample size increases. For example between 1000 and 2000 there will be in general more lightly negative difference then between 2000 and 4000, see appendix A.

The first evidence can be explained by the probabilistic nature of the hypothesis, it is intrinsically quite unlikely that a close to random pdf will turn out linear (or any other simple relation) by sampling. The second evidence can be explained in continuation to the latter, sampling more frequently a given pdf will much more unlikely turn out linear (or any other simple relation). This reinforces the suggested hypothesis.

The third and final type of differences is indicated by very bright blue or red spots, which show a brutal increase or decrease in normalized mutual information. This latter has a less straightforward explanation but is mainly due to features that are most of the time constant. Features that are very rarely non-zero such as the presence of heavy elements (e.g Pt or Rn), noble gases (e.g He) or rare local environments have two undesirable properties to the NMI. First, two rare features (in the sense that they are often zero or any other constant value) have an oscillating MI behaviour with respect to each other. They can both be exclusive with respect to each other (if one is zero the other is nonzero), then suddenly not, or sometimes be related by luck when very few non-zero pairs correspond. This means that the nominator of equation (4.4) changes significantly from one sample-set to another. Secondly, rare features tend to have a very low information entropy. Now, the information entropy appears in the denominator of equation (4.4), which amplifies the previous oscillations. Netto, this results into brutal increases and descreases of the NMI. In summary, this third category of bright spots is caused by rare (sparse) features and the observation of this latter can thus be used to detect them.

4.3 Interactive figures and code availability

This chapter contains many NMI-grams, which are visual illustration of the NMI-matrix between numerous features in form of a heatmap. Unfortunately, the resulting static figures are very dense in amount of features, which limits visibility. It was therefore chosen to include in this written dissertation 'simplified' NMI-grams by the use of coded labels, which ease the visualization. For each of these 'simplified' NMI-grams a corresponding interactive figure is available on the supplementary website of this work, found in Ref. [88]. These interactive figures are highly recommended as they make it easy to zoom, move, and find the corresponding features by simple mouse hovering.

Moreover, the code that was used to obtain the results (including the graphical illustrations) of this chapter can be found on the *github* repository of this work, see Ref. [89].

4.4 Cross-feature information analysis

This section analyzes the 6 cross-feature submatrices, as displayed in figure 4.2.1. First the three intra-module NMIs are presented followed by the 3 inter-module NMIs.

4.4.1 Cross intra-compositional mutual information

It will now be studied how the different *composition* features are related with respect to each other.

Convergence

Before diving into the mutual information, let us first verify convergence of the mutual information between composition-related features with respect the dataset size.

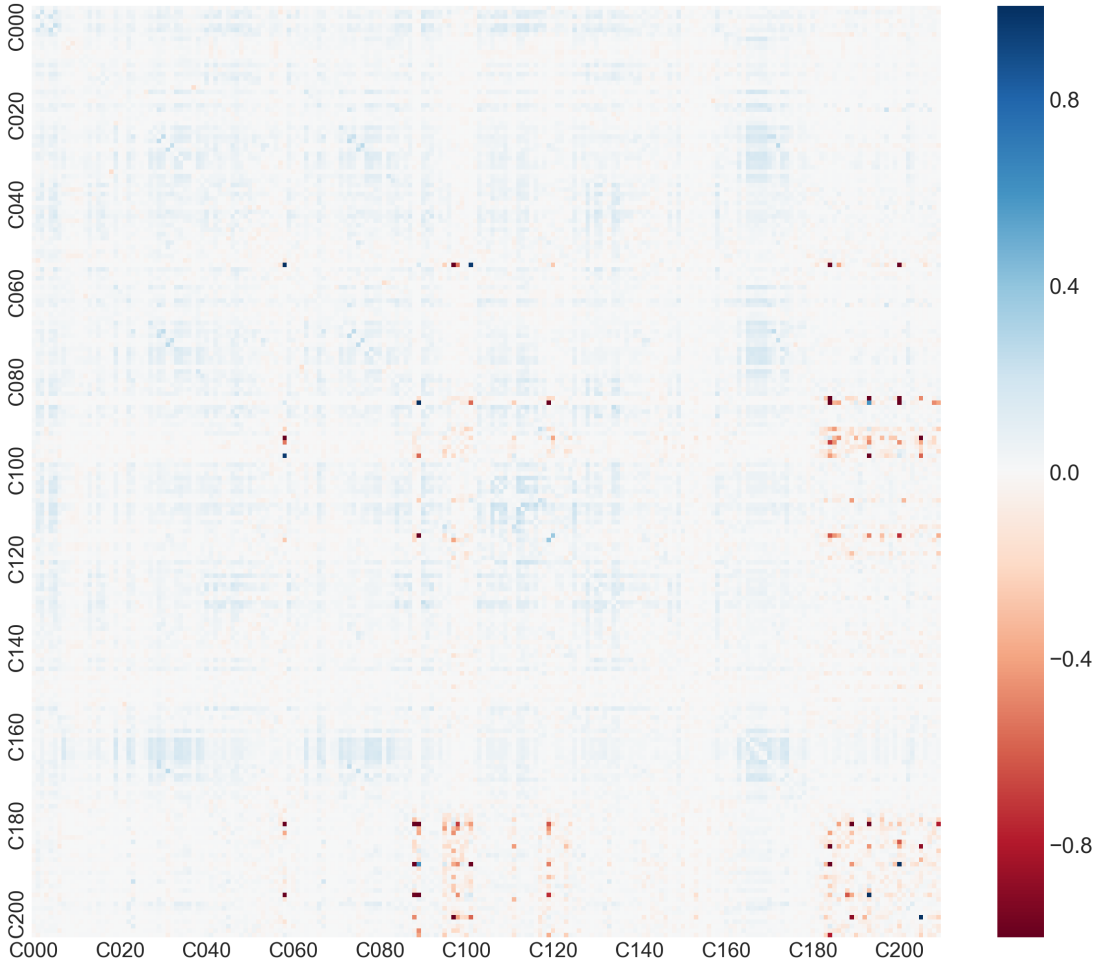

Figure 4.4.1: Convergence NMI-gram for intra-compositional features. Computed by the difference in normalized mutual information between 4000 and 400 samples for composition-related features. A clear increase in mutual information is seen by the bluish zones, while some red points show the presence of sparse features as explained in the text. The features corresponding to the coded labels (**C**Xyz) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

Figure 4.4.1 depicts the difference in mutual information between 4000 and 400 samples, where blueish colors are chosen for positive differences while reddish colors for negative differences. More differences in NMI for different dataset sizes can be found in appendix A.1, where the sequential difference in sizes 200, 400, 600, 800, 1000, 2000, 4000, 8000, 16,000 and 32,000 are taken and shown in an animated figure.

The samples were drawn randomly in the previously described dataset. These dataset sizes were chosen to clearly show the impact of the sample size on the normalized mutual information. First, the three main types of differences (light blue, light red and bright blue/red) can be spotted and can be explained as was done in the general convergence discussion. The most important point to note is that the light positive differences (and thus important relations) are vanishing around 8000 samples (difference between 4000 and 8000).

Convergence is thus reached around 10,000 features. This is confirmed by the animated figure in appendix A.1 that shows almost no difference in NMI between 32,000 samples and 16,000 samples.

In more details, the NMI changes in the beginning strongly for sparse features (thus negative - bright red - and positive - bright blue - differences in NMI, as explained previously) such as any pair of the following set:

- Elemental fraction for less common elements such as He, Ne and Sc elemental fractions.
- Minimum and mode (over the elements) of the orbital filling of less common orbitals (d and f).
- Magnetic moment of the constituent elements, which is itself linked to the previous point.

Then, around 32,000 samples everything has converged except the pairs formed by any element of the set {He, Ne, Be element fraction} with the set {minimum ground sate magnetic moment, minimum f-orbital electrons, minimum number of unfilled s-electrons}, which is again due to their sparse nature.

Finally, and more interestingly, one can see on this same figure (16,000-32,000 difference) three light-blue lines corresponding to the *atomic weight*, *melting temperature of the constituent elements* and *volume per atom* features. This shows that those features are related to all other features but the relation requires a high number of samples to be found, while the NMIs of other features have already mostly converged.

Results

The normalized mutual information for the different pairs of composition-related features are shown in figure 4.4.2, a sequential monochrome heatmap, for 32,000 representative samples. As the x-axis and y-axis contains the same compositional features, the resulting matrix should be symmetric with unit diagonal, which is indeed confirmed by the results.

As this NMI-gram (and the upcoming ones) contain a high amount of information and probably deserve a study on them alone, it is difficult and probably impossible to have an exhaustive enumeration and discussion of all remarkable relations. However care and patience was taken to note most of the more noticeable and remarkable points and will be discussed next.

The heatmap is globally quite whitish and thus means that the cross mutual information is globally high, which shows the fact that the different featurizers are strongly related due to their common compositional nature. In contrast, it will be shown later that the cross composition-structure mutual information is globally much lower. This observation tells us that this compositional feature set is quite redundant, and thus emphasis the need of feature selection.

Next, around the diagonal several whitish squares are seen, which primarily correspond to feature classes (featurizers) that generate a set of strongly related features. For example, the `AtomicOrbital` class generates the features *HOMO character*, *HOMO element*, *HOMO energy*, *LUMO character*, *LUMO element* and *LUMO energy* which are in fact strongly related because they arise from a same underlying chemical element (i.e the element with maximum atomic number which is shown to be related with HOMO/LUMO). The

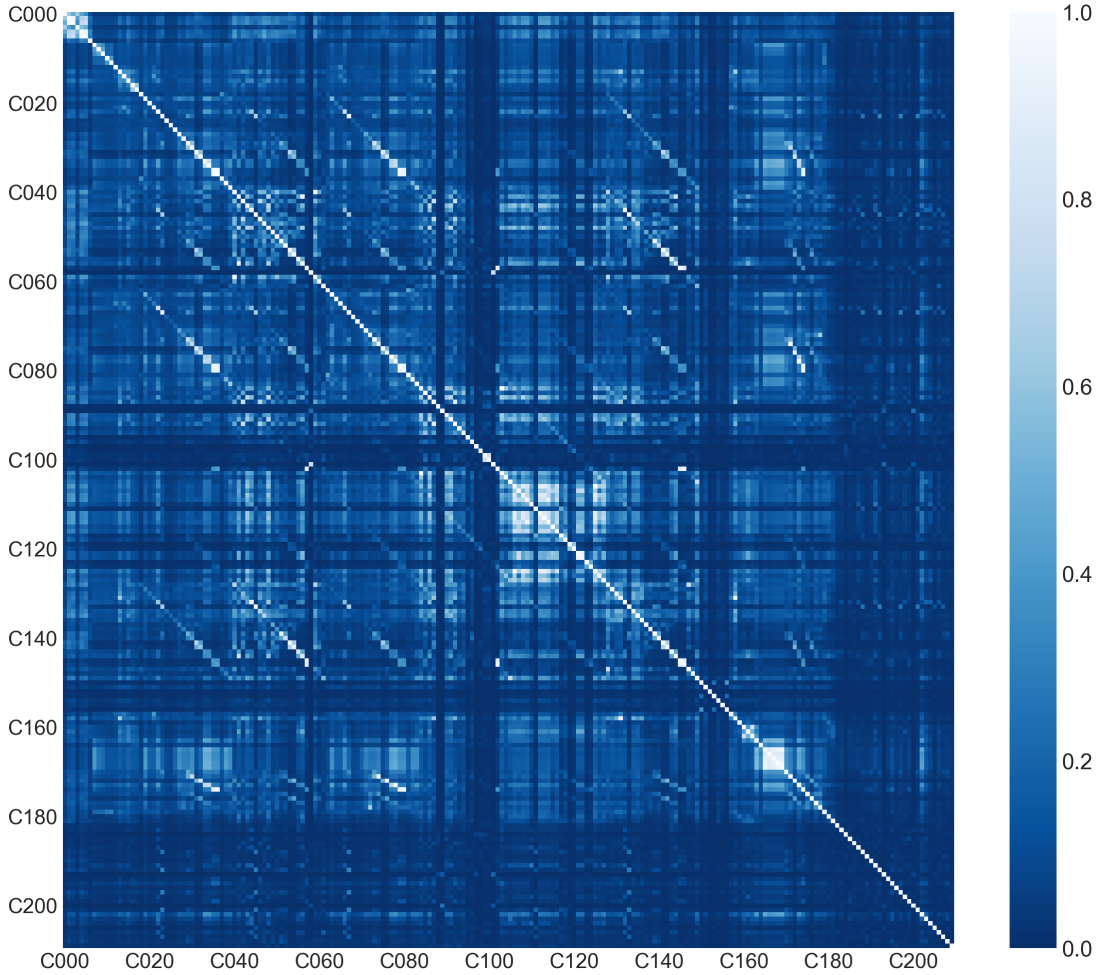

Figure 4.4.2: NMI-gram for intra-compositional features, defined as the normalized mutual information between composition related features. It has been computed from 32,000 randomly selected samples. Whitish zones correspond to a strong inter-relation while bluish zones correspond to a low inter-relation. See text for an in-depth analysis. The features corresponding to the coded labels (**C**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

HOMO/LUMO energy is itself computed from the electronegativity which is in turn related to the element, which altogether explain the high NMI. The same happens for the electronegativity difference (mean, maximum, minimum and standard deviation are strongly related), the atomic packing efficiency (3 or 5 atomic clusters are strongly related), the stoichiometric norms (L-2, L-3, etc. norms are strongly related) and the oxidation state statistics (mean, minimum, etc. are strongly related). This intra-class-dependency behavior is interestingly always present, but much less noticeable for the other classes.

Furthermore, a bunch of parallel white lines can be seen off-diagonal. These corresponds to high NMI's between different statistics of the same property. For example the average deviation is interestingly related with the maximum of the Mendeliev number. The same happens for the following statistics in the left column combined with a property in the right column:

This phenomenon is easy to explain for the electron shells as they fill progressively following

Pair of statistics	**Property**
Maximum and mean	Ground state magnetic moment
Average deviation and mean	Ground state bandgap
Mean and range	Mendeliev number
Maximum and range	Number of unfilled s-electrons
Average deviation and range	Number of unfilled p-electrons
	Number of unfilled d-electrons
	Number of unfilled f-electrons

the *Aufbau principle*, giving for example rise to deviation that grows with the maximum, but less straightforward for the others. Moreover, this phenomenon only appears for the enumerated properties making further investigation interesting.

Still looking very globally on the figure, two dark blue bands can be distinguished covering the NMI-gram corresponding to the features *minimum s, p, d, f orbitals* with all other features for the first band and physical properties (formation energy, bulk modulus among others) with all other features for the second band. The first band is less straightforward to explain but certainly not less interesting: it shows that the *minimum* orbital statistics are globally less important then the corresponding *maximum* statistics. The second band is more easy to explain: it contains macroscopic physical properties (such as formation energy or bulk modulus) and are thus globally less related with other elemental features. They were kept on the NMI-gram to show their relation with the composition. Interestingly the formation energy is shown to be quite related with the composition.

Going more into the details of the NMI-gram, many bright spots indicate the strong relation between compositional features that are often obvious such as the number of unfilled electrons with the number of filled electrons, the column number with the electronegativity, the column with the number of s/p/d-electrons, the covalent radius with the ground state atomic volume, the atomic number with the atomic weight and many more. Similarly, many dark points are unrelated obvious properties such as the column number with the magnetic moment, the column number and the atomic weight among others.

On the other hand some much less obvious related and unrelated features exist. For example, the elemental melting temperature appears to be related to the atomic weight (or number) and the stoichiometry is apparently related to the s, p, d anf f-orbital filling. Also, the presence of most elements are not inter-related, showing that they mostly appear in more or less random combinations, except for some pairs such as Oxygen with Lithium, Oxygen with Phosphor. This is interesting and could potentially be explained by some underlying physical law, or simply due to a bias in the dataset, and thus needs further investigation. Another less obvious and remarkable observation is how the presence of Oxygen really differs from all other elements by having a relative strong relation with many elemental properties such as the compound stoichiometry[2] or magnetic moment but also physical properties such as the formation energy. By looking further, Fluor and Lithium are also very determining elements for many properties, but less than Oxygen.

Finally, it is interesting to note how the different statistics can be complementary or redundant with each other. For example the maximum electronegativity is unrelated with the *maximum* covalent radius, but highly related with the *minimum* covalent radius. This

[2]Probably due to the many existing binary and ternary oxides.

is meaningful since atoms smaller in size are globally more electronegative.

As a conclusion, the mutual information between the different features is globally quite high. Many obvious relations exists but also interesting non-obvious relation were found that deserve a further more in depth investigation. Also, the presence of certain elements were seen to be determining for other compositional attributes. Unfortunately, these high and low mutual information are quite chaocitally distributed among the different classes and features, which makes it difficult to remove or select features *a priori*. This motivates the need of a more systematic and automatic selection algorithm, which will be constructed later.

4.4.2 Cross intra-structural mutual information

We now study how the different *structure* features are related with respect to each other.

Convergence

The difference in normalized mutual information between 400 and 4000 samples can be found in figure 4.4.3, where blueish colors are chosen for positive differences while reddish colors for negative differences. Remarkably, no light blue and very little light red zones are present, which show that the cross-mutual information has indeed converged around 400 samples, which is remarkably low. The dark red and blue points are category 3 points, and thus indicating the rare (i.e sparse) features and should thus be discarded with respect to the aforementioned discussion. In appendix A.2, the animated figure shows us that most of these dark points persist even between 6000 and 8000 samples, enabling us to identify the sparse features. Those sparse features are the binned radial distribution function, below 1.2 Å with all bond fractions except the following bonds,

Al-O, B-B, B-O, Ca-O, Co-O, Cr-O, Cu-O, Cu-S, F-Li, Fe-O, K-O, Li-O, Mg-O, Mn-O, Na-O, Ni-O, O-P, O-Si, O-V, Si-Si.

Physically, this means that the RDF is often zero below 1.2 Å and that only the aforementioned bonds are recurrent (at least in the MP database). For the data scientist, this means that those sparse features should be avoided, because they are prone to overfitting.

Results

The normalized mutual information for cross structural features can be found in figure 4.4.4 computed with 8000 representative samples. The NMI-gram can be divided in three zones: bond fractions with bond fractions, bond fractions with the symmetry features and symmetry features with symmetry features.

The first zone is discarded by the convergence results. The second zone combines rare and unrare features and should thus be carefully interpreted. From the figure, the bond fraction seems to be linked with symmetry properties such as the crystalsystem or the presence of a centrosymmetry, especially for the O-P bond. Moreover it seems to be also linked with elastic properties such as the bulk or shear modulus. Due to the rarity of the bond fraction, these conclusions are only an indication of possible relations with no firm evidence.

The third zone, symmetry features with symmetry features, are safe to interpret and contain much useful information and will be discussed next.

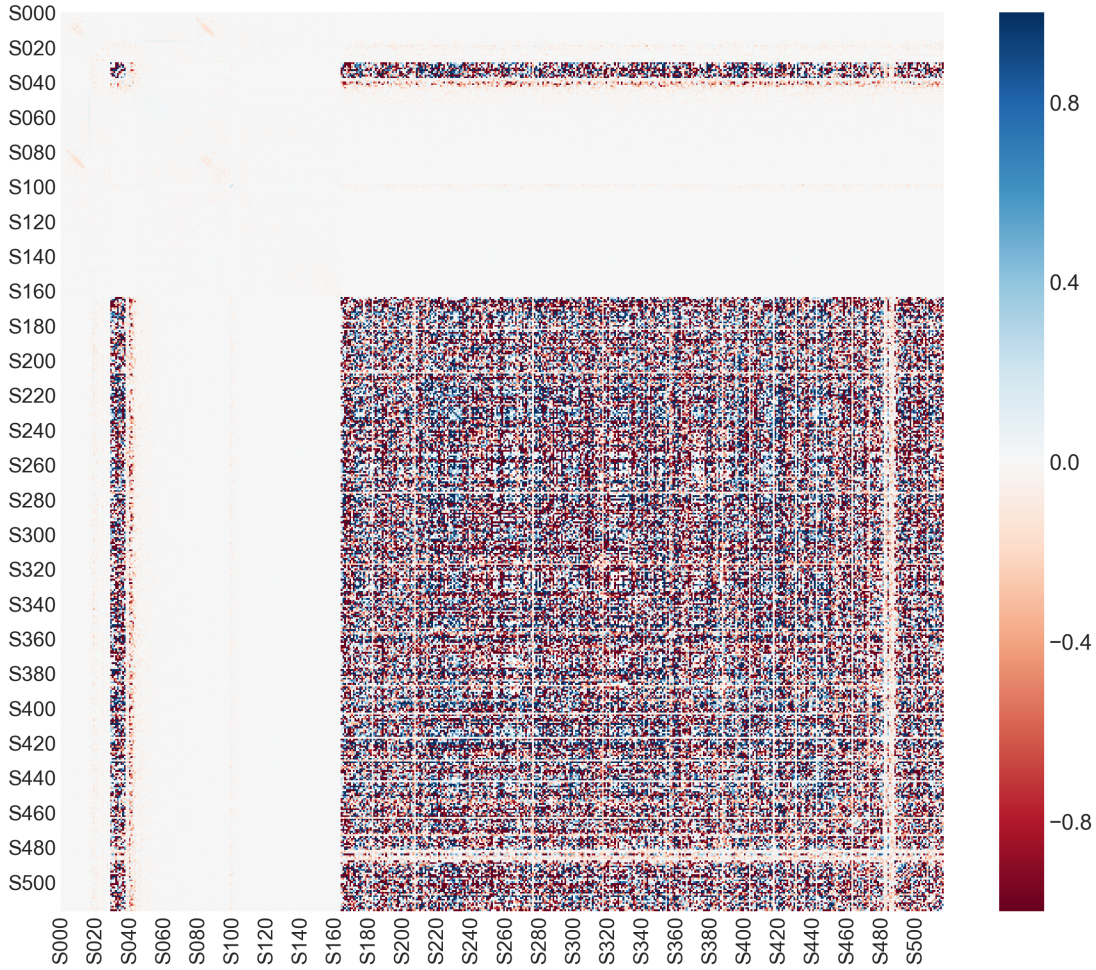

Figure 4.4.3: Convergence NMI-gram for intra-structural features. Computed by the difference in normalized mutual information between 4000 and 400 samples for compositional related features. A clear increase in mutual information is seen by the bluish zones, while some red points show the arrival of sparse features as explained in the text. The features corresponding to the coded labels (**S**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

Around the diagonal, it can be seen that successive eigenvalues from the (sine-)Coulomb matrix are mildly related to each other. For example, the second eigenvalue is mildly related to the first and third eigenvalue, but less with other eigenvalues (the NMI is monotonically decreasing with the inter-eigenvalue distance). This is slightly more pronounced for the sine-Coulomb matrix then the Coulomb matrix. Also, the `chemical ordering` featurizer, producing three features for shell 1, 2 and 3 are in fact closely interrelated as can be seen. The same happens for the `Structural heterogeneity` class for the neighbour distance, relative bond length and cell size. In contrast, the `DensityFeature` class provides a set of complementary, unrelated features.

Furthermore, and more interestingly, the relation between the crystal-system, crystal-system integer (which is basically a mapping of the previous feature), and the space group number is fully retrieved as should be with a NMI of 0.94. However the relation between the space group number and presence of a central symmetry is only partially retrieved with a NMI of 0.36. The presence of a central symmetry should normally be fully deterministic from the

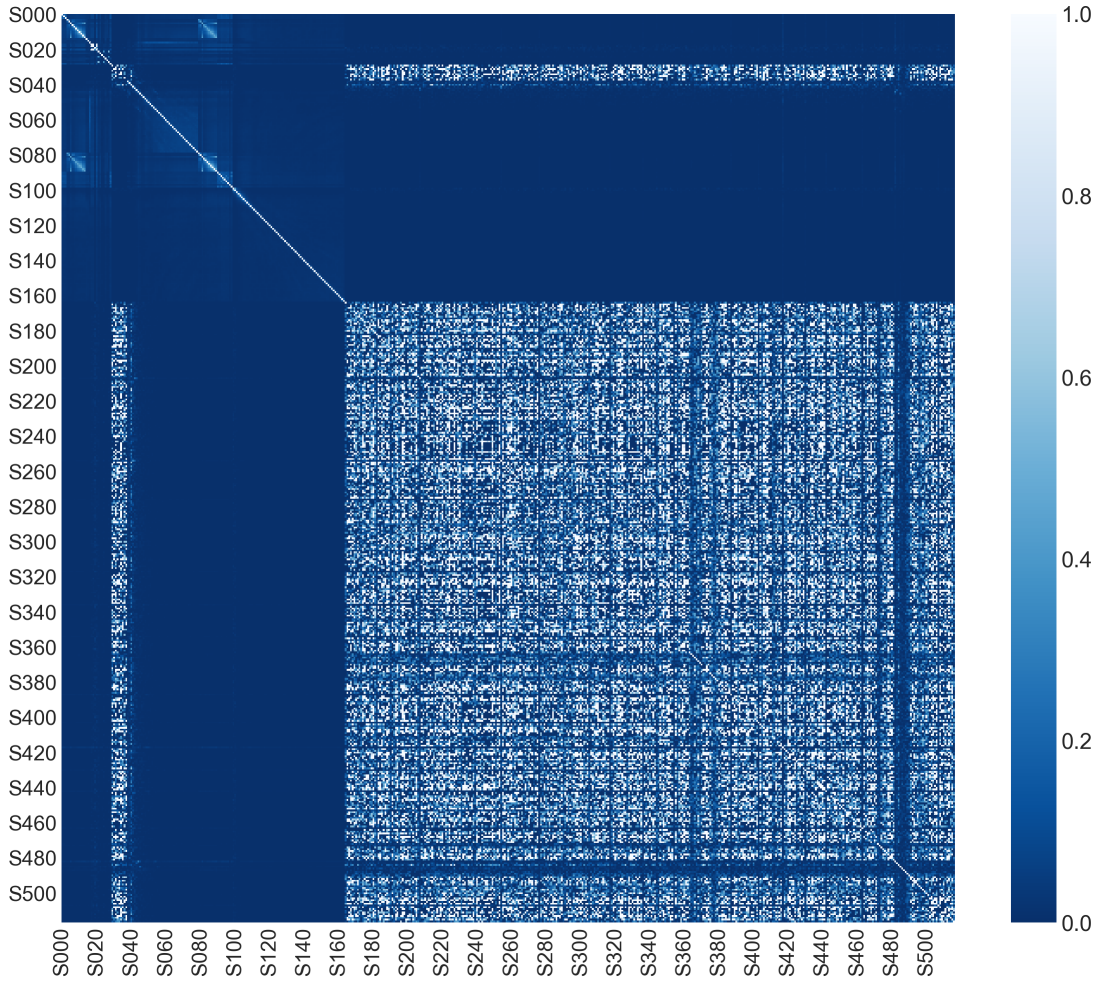

Figure 4.4.4: NMI-gram for intra-structural features, defined as the normalized mutual information between structure related features. It has been computed from 8000 randomly selected samples. Whitish zones correspond to a strong inter-relation while bluish zones correspond to a low inter-relation. See text for an in-depth analysis. The features corresponding to the coded labels (**S**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

space group number, as the corresponding point group states whether a central symmetry ($\bar{1}$) is present or not.

Going farther from the diagonal, other interesting observations can be made. The Coulomb and sine-Coulomb matrix appear to be mildly to strongly related with a NMI of 0.45 for each corresponding eigenvalue. The volume per atom seems slightly related with the RDF (NMI of 0.10), the spacegroupnumber slightly with the sine-Coulomb matrix (NMI of 0.11) and the maximum packing efficiency slightly with the structural heterogeneity (NMI of 0.12). Other descriptor pairs are quite complementary. From this knowledge a good set of complementary descriptors would be the density, crystal-system, presence of centrosymmetry, the packing fraction, the sine-Coulomb eigenvalues 1, 3, 5 and 7 and the RDF at 1.5, 2, 2.5, 3, 3.5 and 4 Å.

4.4.3 Cross intra-site mutual information

It will now be studied how the different *site* features are related with respect to each other. These features account for the local environment of a *particular* site inside the structure. For the present analysis it was chosen to take the mean over all sites in the unit cell, giving a more global descriptor as this is more suited for the upcoming tasks.

Convergence

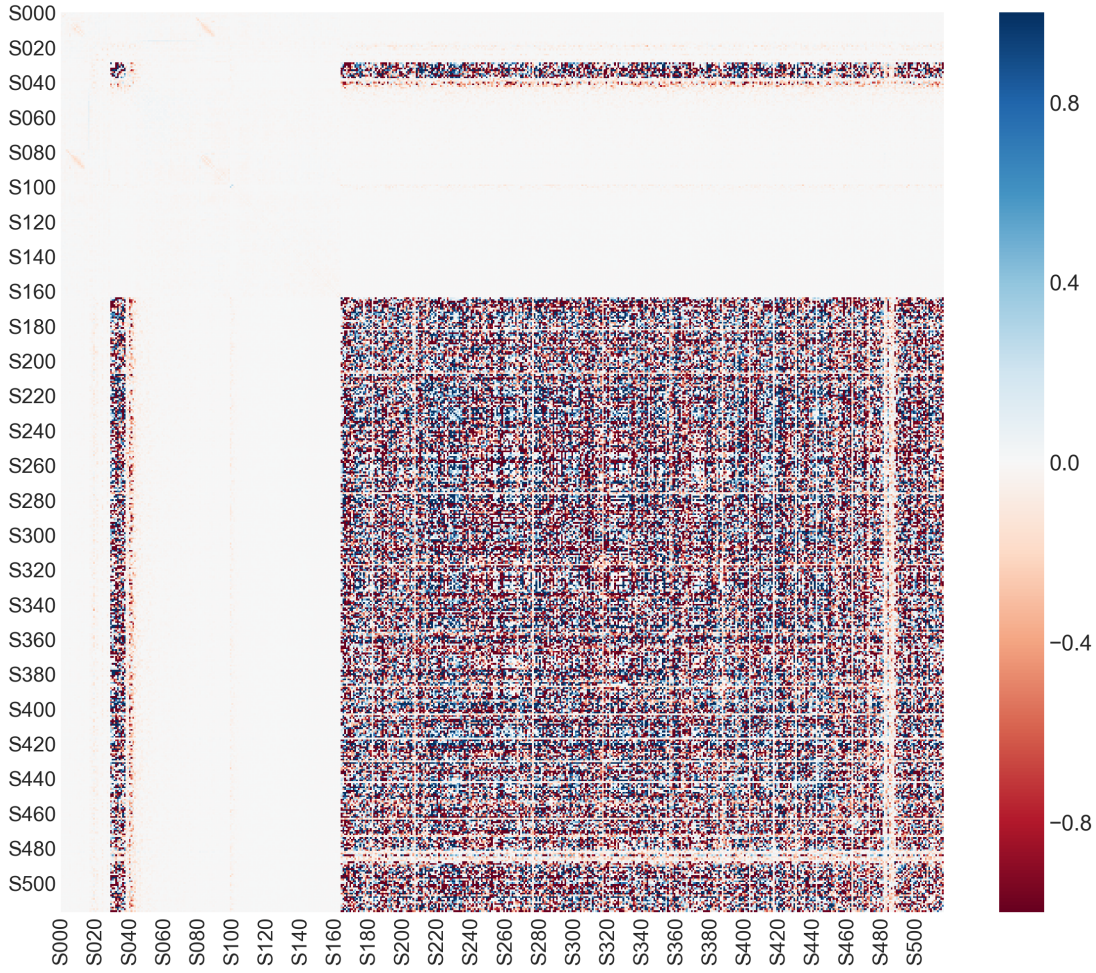

Figure 4.4.5: Convergence NMI-gram for intra-site features. Computed by the difference in normalized mutual information between 8000 and 800 samples for site related features. A clear decrease in mutual information is seen around coordination environments that were initially overfitted. The features corresponding to the coded labels (**L**xyz) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

Figure 4.4.5 depicts the difference in normalized mutual information for site-related features between 800 and 8000 samples. Most of the heatmap has converged at 800 samples, except some pairs around the diagonal and intra-`Chemenv` pairs. The pairs around the diagonal correspond to different local environments with the same coordination number. It should be noted that the difference is principally negative, showing that increasing the number of samples decreases the NMI for these pairs. This can be explained by having mutually exclusive occurring local environments when the sample space is low. When increasing sample space some previously incompatible combinations are seen to be erroneous. Finally

and fortunately, around 4000 samples everything has converged as can be seen in appendix A.3, with absolutely no category three points.

Results

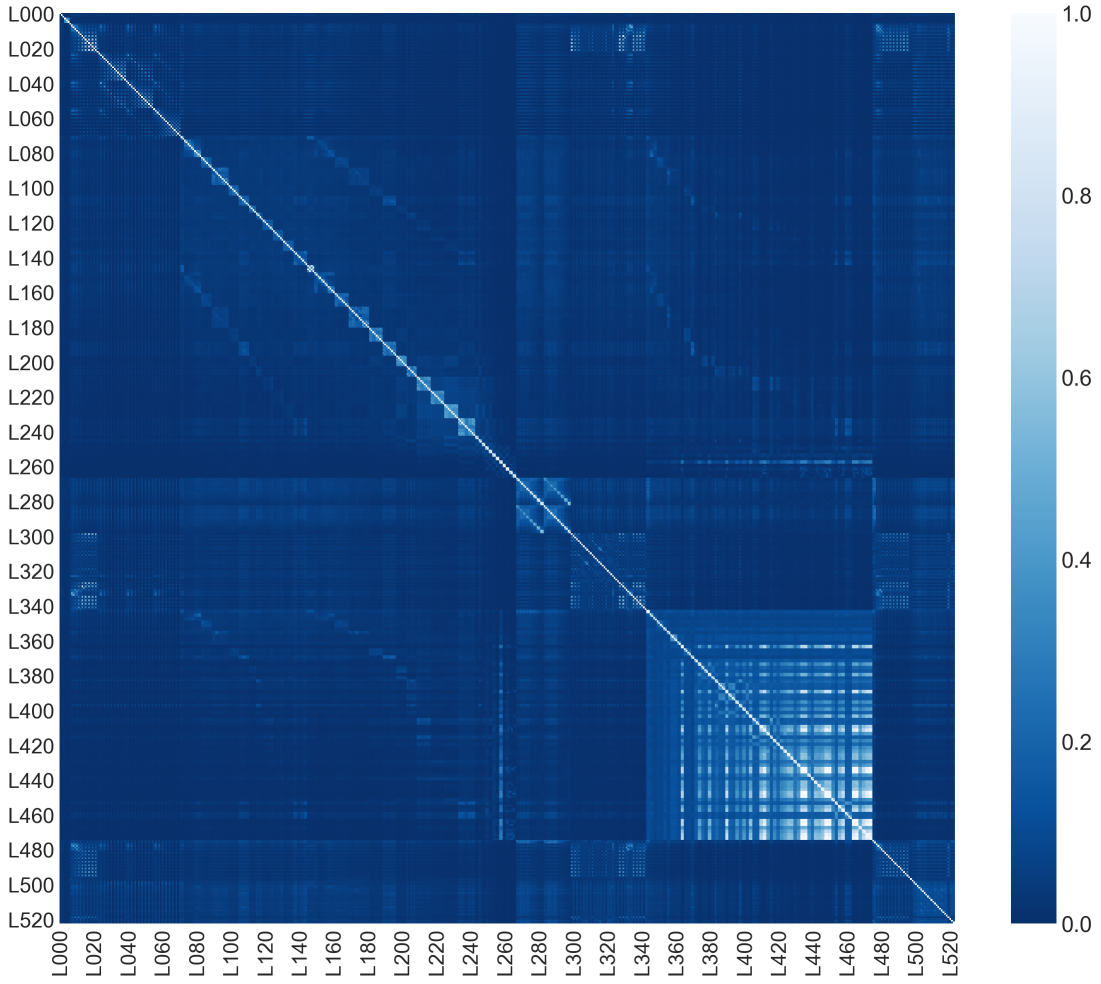

Figure 4.4.6: NMI-gram for intra-site features, defined as the normalized mutual information between site related features. It has been computed from 8000 randomly selected samples. Whitish zones correspond to a strong inter-relation while bluish zones correspond to a low inter-relation. See text for an in-depth analysis. The features corresponding to the coded labels (**L**xyz) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

The final converged cross inter-compositional-structural NMI-gram can be found in figure 4.4.6 for 8000 samples. It is dense in information with many bright areas between classes, showing a clear presence of redundancy. This redundancy is primarily due to the same local environments descriptors (for example trigonal-planar) that exist in different featurizers.

When looking around the diagonal, several observations can be made. First, the mean and standard deviation of `AGNIFingerprint` integrals at successive widths (defined by η), are naturally related by a NMI of around 0.3 in value. Interestingly the standard deviation between AGNIF-integrals with a same width but different directionnality (x,y

and z) seem to be related, but not the mean. Similarly, successive bins of the generalized radial distribution function (GRDF) are related.

Moreover, many squares are seen around the diagonal, which shows that different coordination environments with the same corresponding coordination number are related by a NMI of around 0.2 in value. A possible explanation is that it is unlikely to have different sites with a same CN but different local motif inside a same crystal.

Furthermore, the Voronoi indices (which correspond the n-facets) are shown to be interrelated, which is quite natural as they should be wisely combined in order to fit together as a full polyhedron. Also, the Voronoi area of a cell is related with the Voronoi cell volume (NMI of 0.32) and the Voronoi mean facet distance, as the corresponding means and standard deviations are closely related. The `Chemenv` class shows an interesting behaviour with many bright spots on a rectangular lattice, which is formed by the combinations of two sets of coordination environments that are thus highly related. Two explanations are possible: those structural motifs do often occur together or those structural motifs do *not* often occur together. By analysing those local motifs, which are less common environments such as pentagonal pyramid, end-trigonal-face capped trigonal prism, triangular-face bicapped trigonal prism, heptagonal dipyramid, sphenoid hendecahedron or square cupola among others, it can be concluded that the first explanation is the most probable one.

When looking at the spherical harmonic features, it is seen that they are all mildly to lowly related one another (a NMI of 0.2) and also mildly related to the mean bond angle, but absolutely not to the mean bond length which follows intuition.

Going off-diagonal, relations between the different classes can be observed. Globally, there is a lot of redundancy between corresponding order parameters that are representing the same underlying physics. These redundancies create the many observed off-diagonal parallel bright lines that can be seen on the heatmap. For example the `CrystalNNFingerprint` class and `OPFingerprint` class both contain a 120 degrees bent coordination environment feature, giving rise to a mild but not perfect relation (NMI of 0.11), principally because both features are not weighted in the same way (see previous chapter). The same happens with the `OPFingerprint` class and the `ChemEnv` class, with a NMI of around 0.2. They both differ by the reward given for a certain motif, the first uses a Gaussian reward while the second computes the Euclidian distance. As a consequence `CrystalNNFingerprint` is also related with `ChemEnv` in the same manner.

Redundancy is also seen between radial integrals such as the GRDF, `AGNIFingerprint` class and `GaussianSymmetry` class. For example the GRDF has a NMI of 0.38 with the AGNI integrals. Similarly, the mean average bond length has a NMI of 0.2 to 0.3 with AGNIF integrals or the GRDF. The $G2$ and $G3$ functions from the `GaussianSymmetry` class (figure 3.3.7 from the previous chapter) are especially strongly related with the non-directional AGNI-integrals, mainly because the `AGNIFingerprint` class also uses the $G2$ and $G4$ functions as basis functions.

Finally Voronoi features are seen to be broadly related with all site features with NMIs of around 0.1 in value, and this with all other features. Especially, the $G2$ and $G3$ functions are well related with an NMI of 0.45 with the Voronoi cell volume. The non-directional AGNI integrals are also well related with the mean Voronoi volume.

Note that a bright line can be seen for the `CrystalNNFingerprint` *CrystalNN CN20* feature at the intersection with the `ChemEnv` class. This is simply because both features are rare

and not occuring together.

As a conclusion, the features can be devided in two main categories of site-related features: local integrals and structural motifs. The first is computed based on an integral of the RDF. The second is computed by a measure giving the deviation of a site's local environment from a reference coordination environment. Features belonging to one of these two categories are seen to be strongly related together, while much less with features from the other class.

4.4.4 Cross inter-compositional-structural mutual information

We now investigate how the *composition* features are related with respect to *structure* features. Are these two classes of properties independent or could one give indications to the other?

Convergence

Figure 4.4.7 shows the difference in NMI between 1000 and 8000 samples for compositional features on the y-axis and structural features on the x-axis. As expected the NMI for pairs of rare features give rise to category 3 points, which have thus not converged, even at 8000 samples. These pairs are simply the combination of rare features that were already identified before, i.e. any combination of the set {bond fractions, RDF below 1.2 Å} with the set {element fractions, s-/f-orbital filling}. All other NMIs have indeed converged, and this already for 1000 samples as can be deduced from the light blue/red zones on other differences in appendix A.4.

4.4.5 Results

The results of this cross-composition-structure NMI can be found in figure 4.4.8 for 8000 samples. The prominent white vertical lines should not be interpreted as they have not converged from the previous discussion. Other zones in the heatmap can however safely be interpreted. From the left upper part of the figure it can be seen that the composition and structure are in fact not totally unrelated as one could think. Especially, the Coulomb matrix shows a remarkable relation with various elemental properties. More specifically the Coulomb matrix shows close relation with the mean atomic number, maximum atomic number, electronegativity and atomic weight, with a NMI of 0.41 between the first eigenvalue and the atomic number. This observation comes from the intrinsic definition of the Coulomb matrix that takes into account the atomic charges, and thus elemental properties. Therefore, the Coulomb matrix can be seen as a descriptor taking both elemental and structural information.

In fact, the upper left part of the heatmap also shows that elemental properties are in fact commonly interrelated, as bright spots always come in form of horizontal bars (and not isolated spots) covering almost completely all elemental features. This confirms what was described before, i.e. that most elemental features are globally very similar and thus mostly redundant.

Next to the Coulomb matrix, the space group number also shows to be closely related with elemental features, which is more surprising. For example, the spacegroup number with

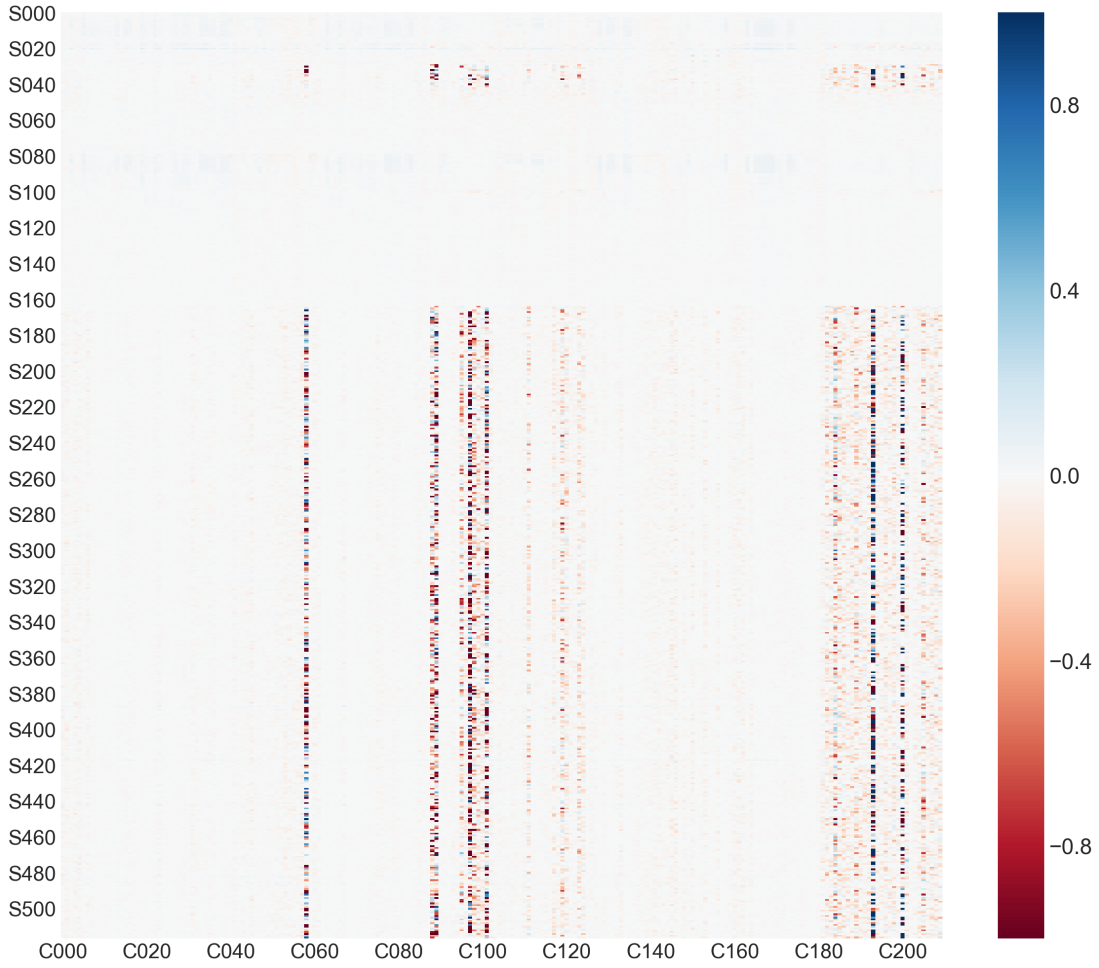

Figure 4.4.7: Convergence NMI-gram for inter-compositional-structural features. Computed by the difference in normalized mutual information between 1000 and 8000 samples for composition versus structural related features. Most features have converged except pairs of rare features such as bond fractions and element fractions (see text). The features corresponding to the coded labels (**C**XYZ and **S**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

(i) the minimum row has a NMI of 0.2, (ii) the stoichiometry a NMI of 0.2 and (iii) the p-orbital filling a NMI of 0.13 in value. This means that some elements (or families of elements) tend to favour some specific structures. In addition, the packing efficiency feature appears to be related also to the elemental features.

It is also seen that most bond fractions (i.e. a pair of elemental species) are not related with the mean elemental properties, which more or less confirms the fact that most species occur independently. There are however exceptions for bonds counting oxygen such as Li-O or Fe-O, as they show bright lines in the NMI-gram. This could be explained by the higher likelihood of binary oxides compared to ternary, quaternary oxides (at least in the considered database), meaning that knowing the (unique) bond implies knowing the composition.

Finally, it is interesting to note that from all the converged element fraction, oxygen shows a non-zero relation with structural properties, such as a NMI of 0.1 with the space group number, and thus appearing to have a slight structural preference when forming

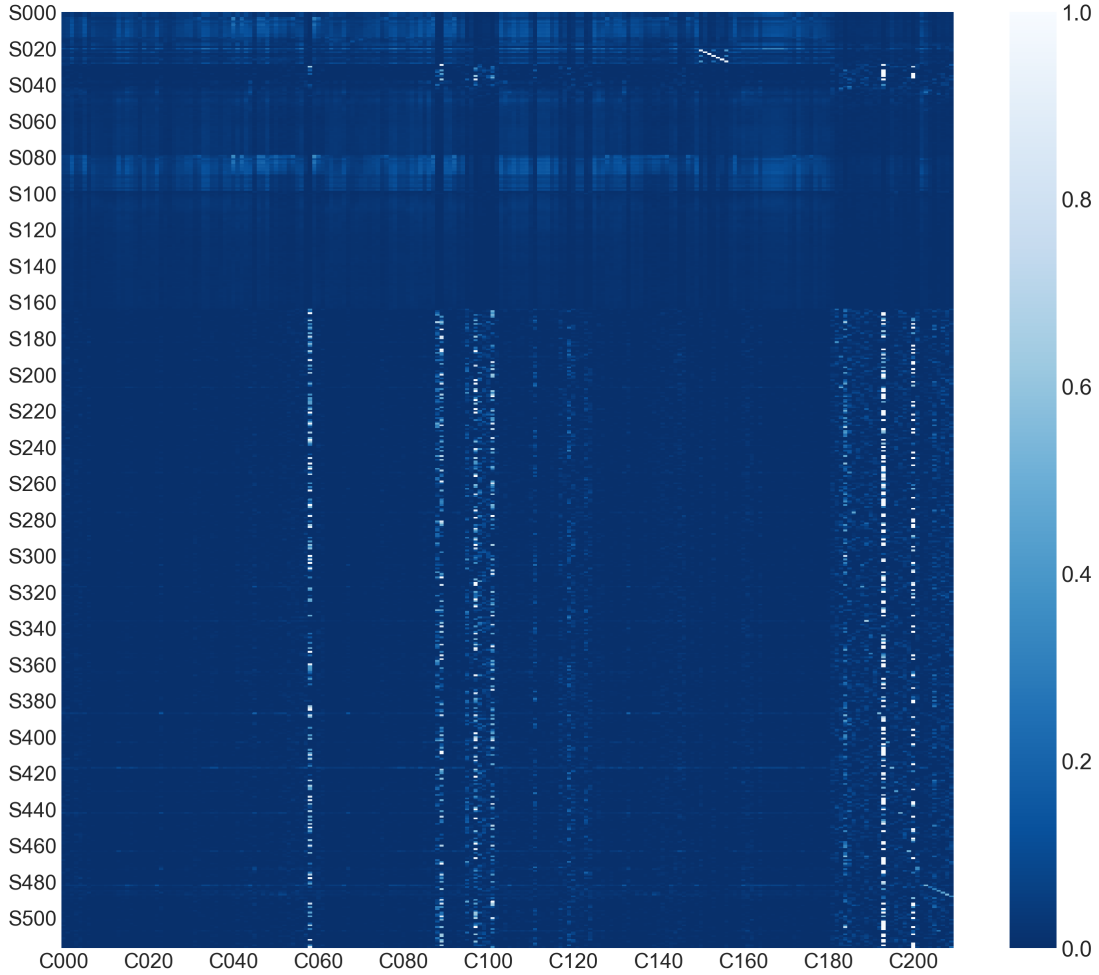

Figure 4.4.8: NMI-gram for inter-compositional-structural features, defined as the normalized mutual information between composition and structure related features. It has been computed from 8000 randomly selected samples. Whitish zones correspond to a strong inter-relation while bluish zones correspond to a low inter-relation. See text for an in-depth analysis. The features corresponding to the coded labels (**C**XYZ and **S**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

crystals.

4.4.6 Cross inter-site-compositional mutual information

Here, the relation between site-specific features and composition features is investigated. In particular, this could give indications if some elements tends to favour a particular coordination. However, we have to keep in mind that both descriptors are averages over the compound, resulting in a 'mean field' analysis.

Convergence

Figure 4.4.9 shows the difference in NMI between 1000 and 4000 samples for site related features on the y-label and composition related features on the x-label. As can be expected, at the intersection of rarely occuring local environments on the y-axis and element fractions, s-, f-orbital filling or elemental magnetic moments on the x-axis occur category 3 points

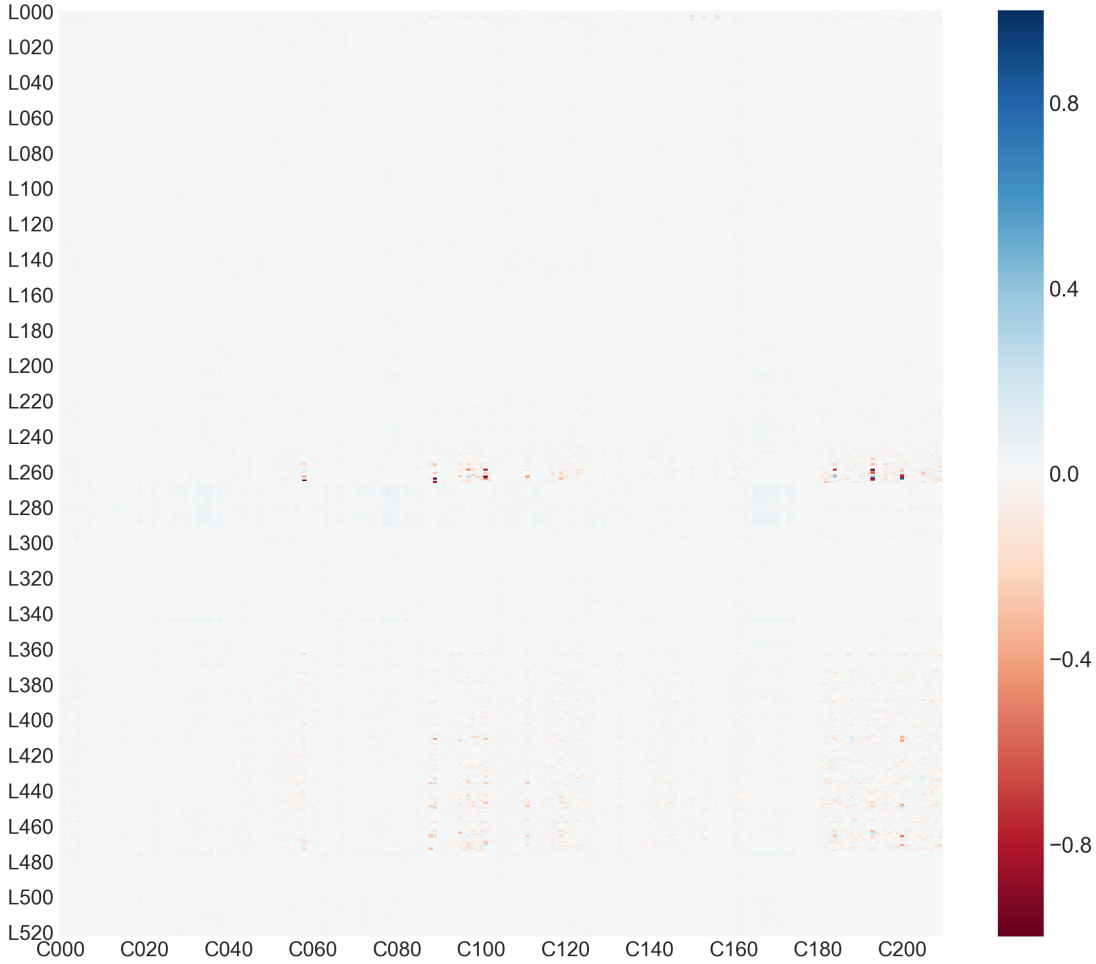

Figure 4.4.9: Convergence NMI-gram for inter-site-compositional features. Computed by the difference in normalized mutual information between 1000 and 4000 samples for site and composition related features. The features corresponding to the coded labels (**L**XYZ and **C**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

that vanish slowly by increasing the sample size. However, they are still present at 8000 samples, final size of the NMI-gram, as can be seen in appendix A.5. Fortunately, all other pairs of features have effectively converged already from 400 samples, as can be seen from the same appendix.

Results

The result of this cross site-composition NMI can be found in figure 4.4.10 for 8000 samples.

The first point to note is that the two categories of features (site and composition) are globally relatively lowly related with NMIs of around 0.1, with some exceptions. This is not surprising, as both categories of features take the average over the unit cell, which means that we compare a mean local environment with a mean elemental composition which can vary considerably with different unit cells. Still, some tendencies are found.

Generally, all site-related distance-based measures are somewhat related with the elemental

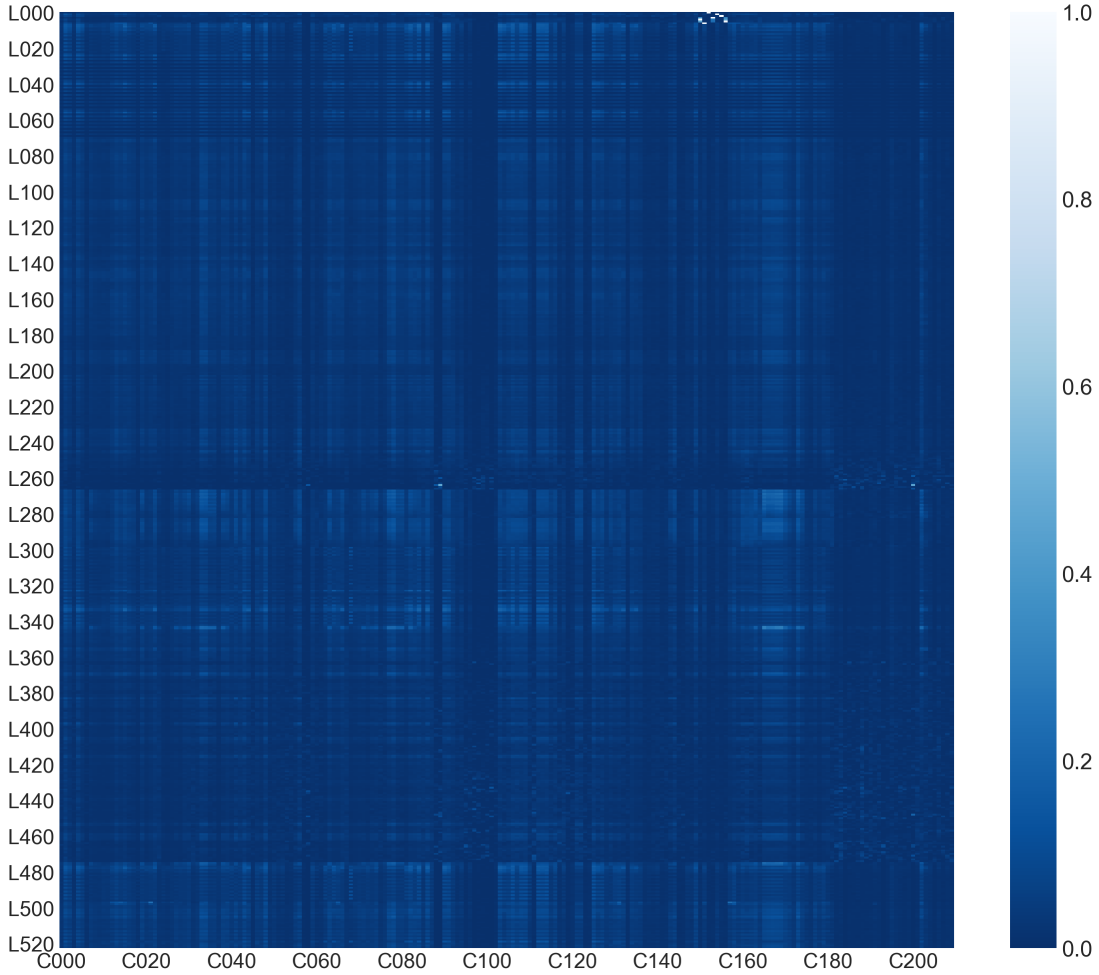

Figure 4.4.10: Cross site-composition NMI-gram, defined as the normalized mutual information between site and composition related features. It has been computed from 8000 randomly selected samples. Whitish zones correspond to a strong inter-relation while bluish zones correspond to a low inter-relation. See text for an in-depth analysis. The features corresponding to the coded labels (**L**XYZ and **C**XYZ) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

composition. More precisely, the Voronoi mean minimum distance or the $G2$ function shows a NMI of up to 0.2 with most elemental properties, with for example a NMI of 0.16 between the mean minimum Voronoi distance and mean periodic row.

Local coordination environment related features (such as the `ChemEnv` class) show a lower relation with elemental properties with a maximum observed NMI of 0.10 in value.

Moreover, the *linear* coordination environment shows a relative high (NMI of 0.2) relation with mean compositional properties, which shows that this motif is restricted to specific compositions. Similarly, from the converged elemental fractions, oxygen and phosphor have their own characteristic relationship with the local coordination environments. These elements are thus more commonly found in particular motifs (e.g oxygen shows a peak for the octohedral environment). As most elements have not converged and all features are averages among all sites, no further analysis will be made. More information about statistics on local environments and its correlation with elements in oxides can be found in Ref. [75].

Finally, the most bright area of the figure corresponds to the relation between Voronoi indices (n-facets) and the stoichiomtry, which are thus shown to be related.

4.4.7 Cross inter-site-structural mutual information

This last cross-NMI study covers the relation between *site* and *structure* features, which one could expect to be related.

Convergence

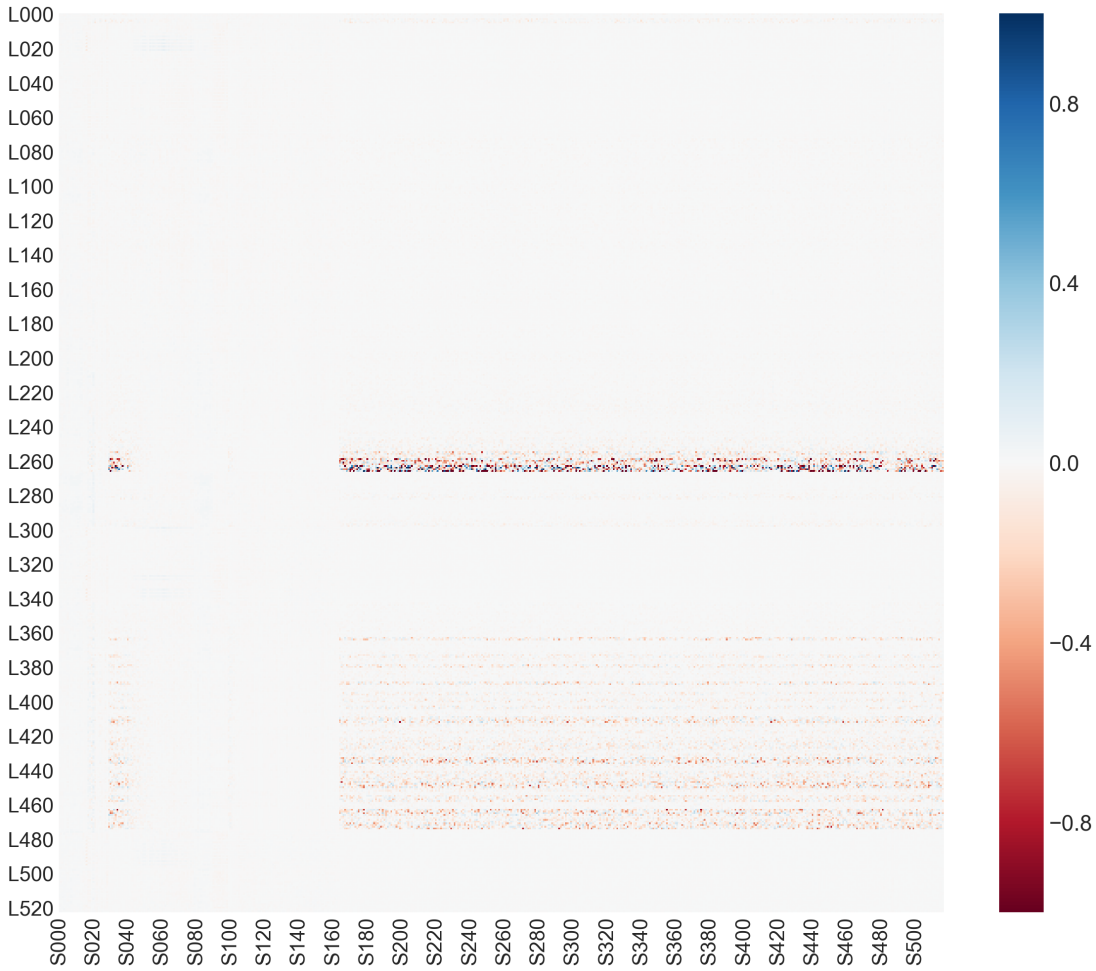

Figure 4.4.11: Convergence NMI-gram for inter-site-structural features. Computed by the difference in normalized mutual information between 1000 and 4000 samples for site related features. Most zones have converged except pairs at the intersection of rare local environments with bond fractions. The features corresponding to the coded labels (**L**xyz and **S**xyz) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

Figure 4.4.11 shows the difference in NMI between 1000 and 4000 samples for site related features on the y-axis and structural related features on the x-axis. Again, at the intersection of rare features, no convergence is reached, corresponding to rare local coordination environments with bond fractions or the RDF below 1.2 Å. Moreover, the difference is mainly negative meaning that when one increases the sample size the NMI between bond

fractions and local coordination environments will actually decrease. Most of those relations will therefore probably disappear if one goes even higher in sample size.

All other regions converge fast, with almost no change beyond 800 samples, shown in the animated figure in appendix A.6.

Results

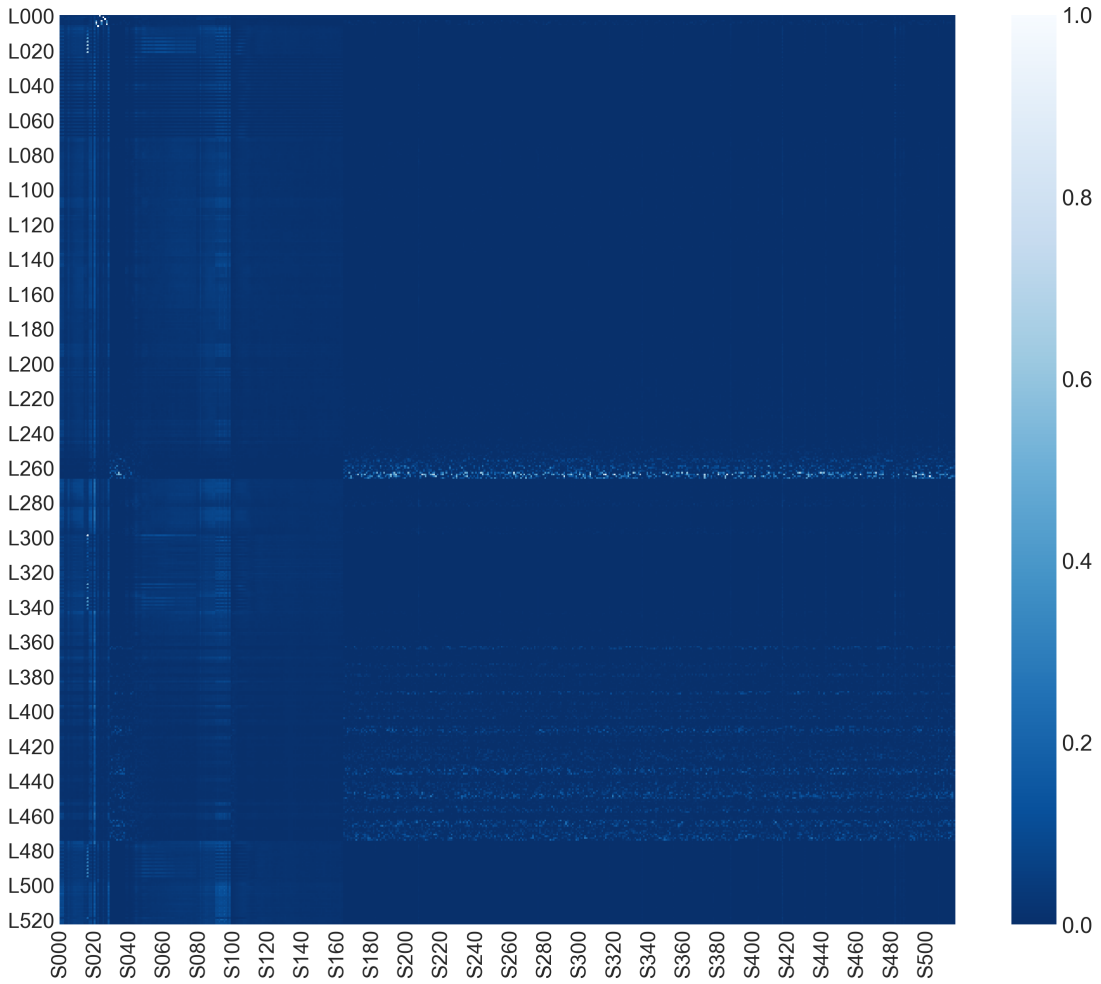

Figure 4.4.12: NMI-gram for inter-site-structural features, defined as the normalized mutual information between site and structure related features. It has been computed from 8000 randomly selected samples. Whitish zones correspond to a strong inter-relation while bluish zones correspond to a low inter-relation. See text for an in-depth analysis. The features corresponding to the coded labels (**L**xyz and **S**xyz) can be found in appendix D while an interactive version of this figure is available in Ref. [88].

The very final cross-result of this NMI study can be found in figure 4.4.12, representing the NMI between site related features on the y-axis with structure related features on the x-axis, for 8000 samples.

One would expect a close relations between the structure and local coordination environment as the first defines the second. This is indeed confirmed by the bright right side of the figure, which relates structural features (except sparse bond fractions) with the coordination environment. Especially, one of the brightest vertical lines corresponds to the relation of

the space group number with both coordination environments (motifs) and radial integral features (based on the RDF) with NMIs from 0.1 to 0.26. The same happens for the structural heterogeneity feature set. This clearly shows that the relation between the structure and local motifs are indeed retrieved.

From the left side of the figure, one can see that most bond fractions are unrelated with specific local coordination environments, which is probably caused by taking the average over sites which washes out elemental characteristics. An exception is found at bonds containing oxygen, such as Li-O or O-P, which show a preferential local coordination environment. This latter probably suffers less from the average as the database contains many binary oxides.

As a final comment, a very high relation (NMI of 0.7) is found between the AGNI features and volume per atom feature, which is quite natural as both can be derived from the RDF.

4.5 Feature-target information analysis

The main task of machine learning models is to predict a particular target as a function of a set of features. In our case, we want to predict material properties, such as melting temperature or formation energy, as a function of the aforementioned list of material features (see chapter 3 for an exhaustive list of all the features). From the vast amount of available features (more than 2000) one could arguably ask which features are really important, in the sense that they are somehow related with the target output. In an attempt to answer this question, this section will present the normalized mutual information for all available features. Features with a high NMI can be seen as physical properties that dictate that particular target, in the sense that the target can be (partially) deduced from that feature (or more strictly that the information entropy can be reduced by the knowledge of that descriptor). They are therefore important if one wants to build a predictive model. In the present work, particular interest is put on predicting thermodynamic properties of solids. Therefore, the target variable is arbitrarily chosen to to be the vibrational entropy at 300 Kelvin. To make the study more complete and show the variability in relevant features, a comparison is also made with the formation energy as a second target.

4.5.1 Vibrational entropy at 300 K

Here we will tackle the problem of finding decriptors that have the highest relation (i.e. highest NMI) with the vibrational entropy of a compound at 300K. A physical introduction to this quantity was given in chapter 1.

Composition

The NMI between composition-related features and the vibrational entropy at 300 Kelvin varies broadly between 0.11 at best and around 0 at worst. This shows the complexity of the problem, which is not easily linked to simple defining properties. From the 279 compositional features, the 20 features with the highest score are shown in figure 4.5.1 with the corresponding NMI score. The right-side of the figure shows the difference in NMI between 1245 and 800 features for convergence purposes. Both barplots have the same range, which ease a visual comparison. Here, the differences in NMI are small compared to

the actual values, but not small enough to guarantee convergence. For example the NMI for the mode atomic number is still growing at 25% between 800 and 1245 samples. However, due to a lack of a larger database and the although acceptable discrepancy in NMI for the different samples, the results will still be used.

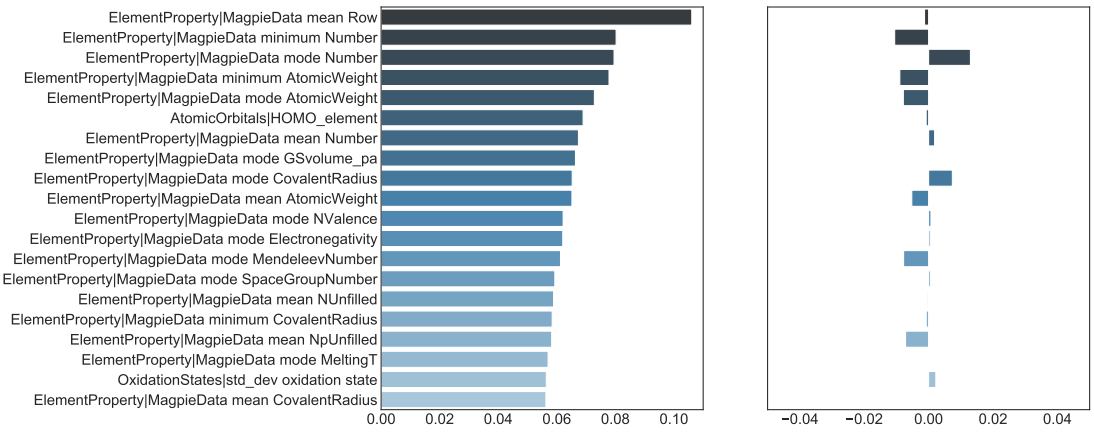

Figure 4.5.1: *Left:* 20 highest ranked *composition* features according to their NMI with the vibrational entropy at 300K. *Right:* Corresponding convergence value, computed as the difference in NMI between 1245 and 800 features. Note that both subplots have the same NMI-range.

It appears that the most important features mainly originate from the `ElementProperty` class, describing the elemental distribution of the composition. In particular, there is one feature well above the others: the mean row in the periodic table of the constituent elements, with a NMI of 0.11. Physically, the periodic row is based on the principal quantum number, which mainly determines the radius of the atoms. It is thus in a sense related to bond lengths in solids, which is known to be important for phonon related properties. Other elemental properties are slightly less important, decreasing marginally and steadily in NMI. It is interesting to see that there are many features with an equal order of importance, and thus no subset coming out from the others. This can principally be explained by the high redundancy among the elemental features. For example, the minimum atomic number and minimum atomic weight are both present at the top of the list, and are known to be both highly related.

Less important compositional features are the individual elemental fractions, solid solution properties (see `YangSolidSolution` class), ionic properties and magnetic moments as can be seen from figure 4.5.2. Most importantly, most of these features have not converged with respect to the sample size due to their sparsity, which adds a reason more to discard them as they are prone to overfitting.

Structure

There are 1245 structural features that have a NMI between 0.10 at best and 0 at worst with the vibrational entropy. Out of these 1245 features the 20 with highest NMI are listed in figure 4.5.3. The right side shows the difference in NMI between 800 and 1245 samples, which is near to zero for all listed features. This means that convergence is reached at 1245 samples and the results are thus safe to interpret.

From these results, we can see that two or three features are really coming out with a

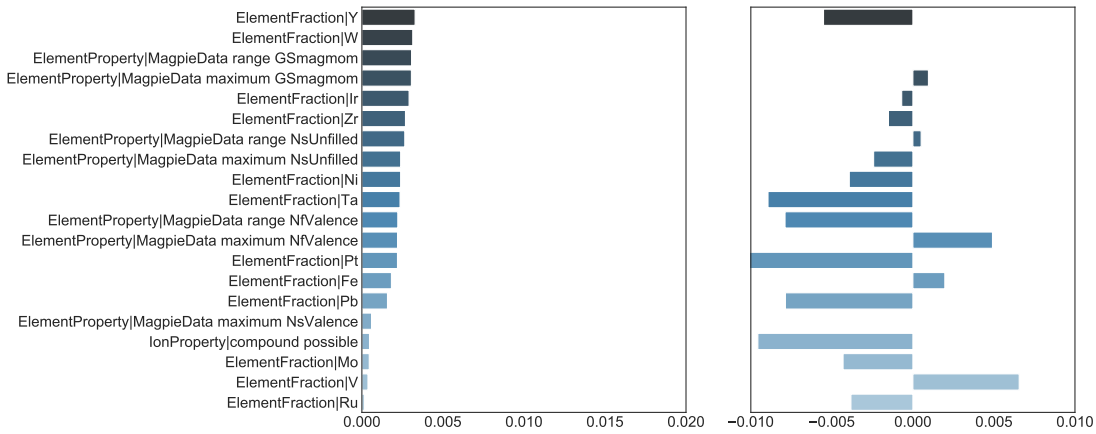

Figure 4.5.2: *Left:* 20 lowest ranked *composition* features according to their NMI with the vibrational entropy at 300K. *Right:* Corresponding convergence value, computed as the difference in NMI between 1245 and 800 features. Note that both subplots have the same NMI-range.

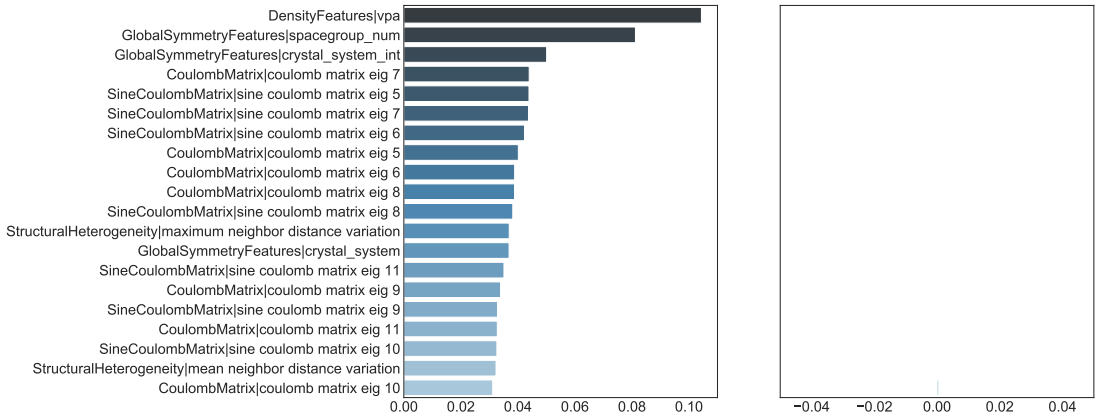

Figure 4.5.3: *Left:* 20 highest ranked *structure* features according to their NMI with the vibrational entropy at 300K. *Right:* Corresponding convergence value, computed as the difference in NMI between 1245 and 800 features.Note that both subplots have the same NMI-range.

significant higher NMI than others, which are the volume per atom feature, space group number and crystal system. The volume per atom is closely related to the packing of the atoms and the interatomic distances, which could be linked to the earlier identified mean row feature. For every Coulomb matrix eigenvalue, there is the same corresponding sine-Coulomb matrix eigenvalue, which is not surprising with regard to the preceding cross-information study. Next to the density and symmetry features, the Coulomb eigenvalues show to be an excellent representation for our target variable. Moreover, a slight preference is given to higher order eigenvalues, as small order eigenvalues appear later in the ranked list.

Figure 4.5.4 represents a similar representation for the last 20 features in our ranked list. As before for the element specific features, here bond fractions, are seen to be less information-full. High angle diffraction values are also present in the tail of the ranked list. Bond fractions are moreover not always converged with respect to the sample size and therefore not of much interest for a predictive model using this database.

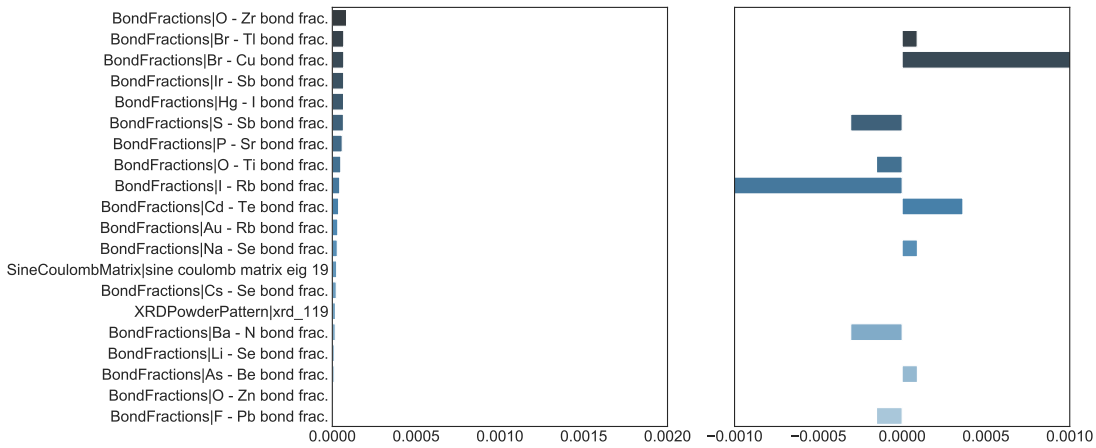

Figure 4.5.4: *Left:* 20 lowest ranked *structure* features according to their NMI with the vibrational entropy at 300K. *Right:* Corresponding convergence value, computed as the difference in NMI between 1245 and 800 features.Note that both subplots have the same NMI-range.

Site

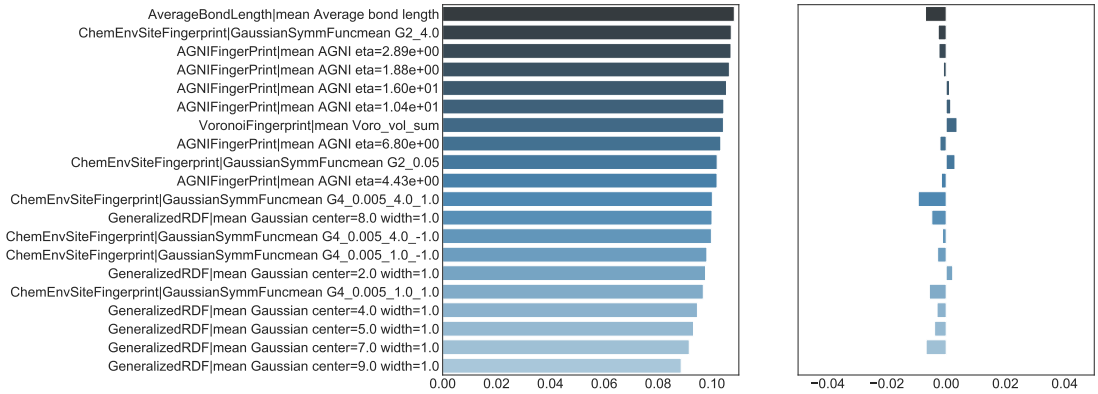

Figure 4.5.5: *Left:* 20 highest ranked *site* features according to their NMI with the vibrational entropy at 300K. *Right:* Corresponding convergence value, computed as the difference in NMI between 1245 and 800 features.Note that both subplots have the same NMI-range.

As our final class of features, the site related features were also ranked following their NMI with the vibrational entropy, and the best 20 out of 517 features are shown in figure 4.5.5. Again, the difference in NMI between 800 and 1245 samples is shown in the right side. These differences are not yet reaching zero around 1245 samples, but small enough to consider the head of the ranked list valid.

It can be seen that these 20 best features (in the sense of having the highest NMI with the target) all roughly have the same NMI, with no clear preference for a particular set of features. This can only be explained by their common underlying nature. They are in fact all derived from the radial distribution of neighbor atoms, or in other words the RDF. For example, the average bond length can be deduced from the RDF as well as the different radial integrals (AGNI, Gaussian symmetry integrals, etc.) presented before. Surprisingly, local coordination environments such as local polyhedron motifs are much less important. Thus, from the two categories of local fingerprints (RDF based and motif based) identified

before, the RDF-based features are shown to be the most important for the vibrationnal entropy at 300K.

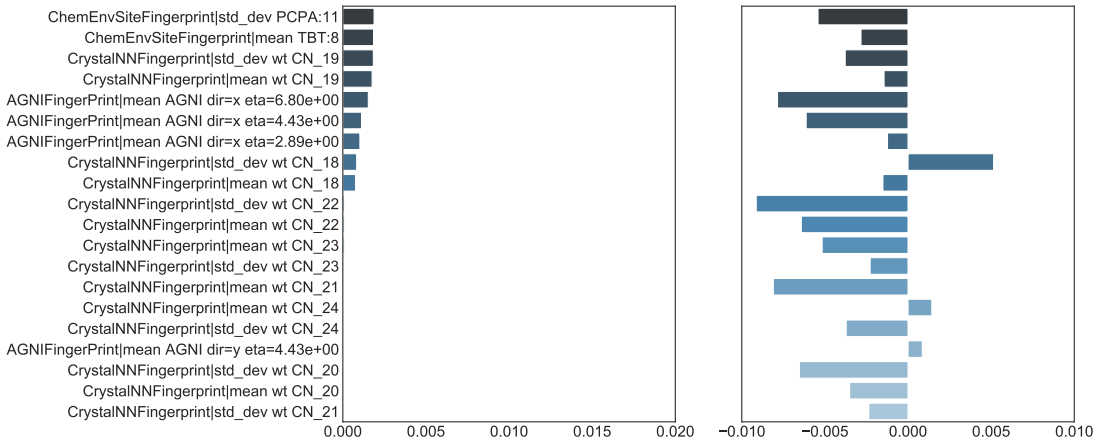

Figure 4.5.6: *Left:* 20 lowest ranked *site* features according to their NMI with the vibrational entropy at 300K. *Right:* Corresponding convergence value, computed as the difference in NMI between 1245 and 800 features.Note that both subplots have the same NMI-range.

The tail of the ranked list is represented in figure 4.5.6, which contains rare local coordination environment and directional fingerprints. Again, they are in fact not converged, which make them delicate to discuss, and are prone to overfitting.

4.5.2 Formation energy

As a final study for this feature-target information analysis section, the same rankings will be presented but this time for the formation energy of crystal structures. As it name suggests, the formation energy represents the energy that is required (or released) when forming the compound from the isolated constituent atoms. As will be seen, the physical descriptors are quite different from the vibrational entropy. The formation energies were provided by the same dataset as was used for the cross-NMI study, see section 4.2.2.

Results

The NMI were computed from 32,000 samples, and are all converged with respect to the sample size. Results are shown in figures 4.5.7, 4.5.8 and 4.5.9.

First, it is important to note that the magnitude in NMI is much higher for composition-related features than structural or site related features. In particular, among the compositional features, the electronegativity (and related ionic character) appears to play an important role in the formation energy. Most structural features are much less related except the packing efficiency and Coulomb matrix eigenvalues, which are in fact closely related with elemental features. The same happens for site related features: most head ranked features are radial features that have some roots in elemental properties.

This example showed that the wide variety of features do not all have the same importance with respect to the target variable. Depending on what is wished to be predicted, some features are shown to be more relevant than others. For example, the vibrational entropy tends to be more related to structural properties, meanwhile the formation energy is more

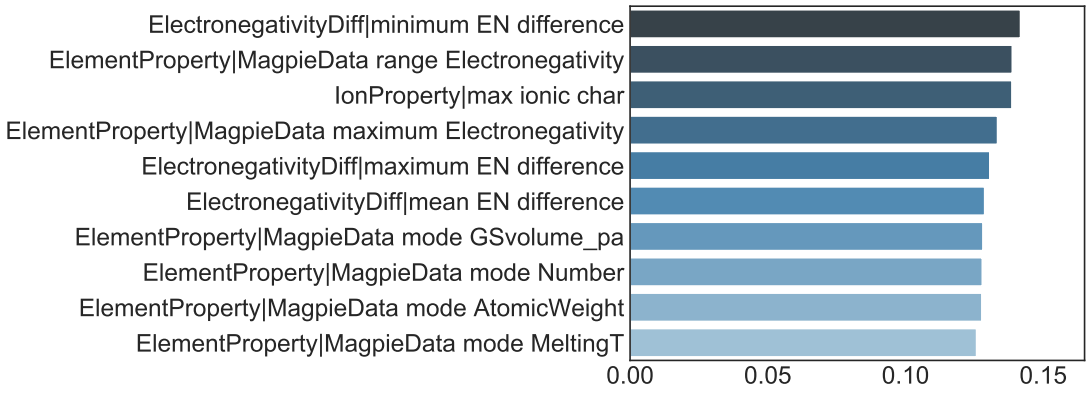

Figure 4.5.7: 10 highest ranked *composition* features according to their NMI with the formation energy.

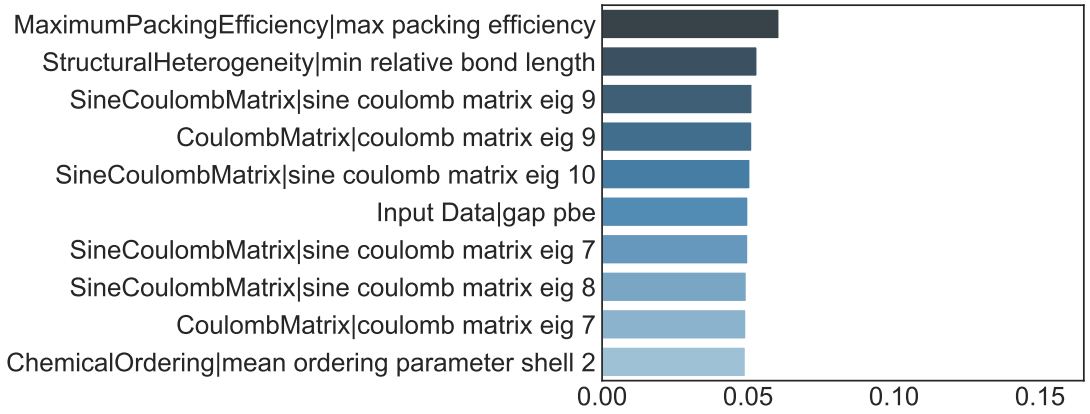

Figure 4.5.8: 10 highest ranked *structure* features according to their NMI with the formation energy.

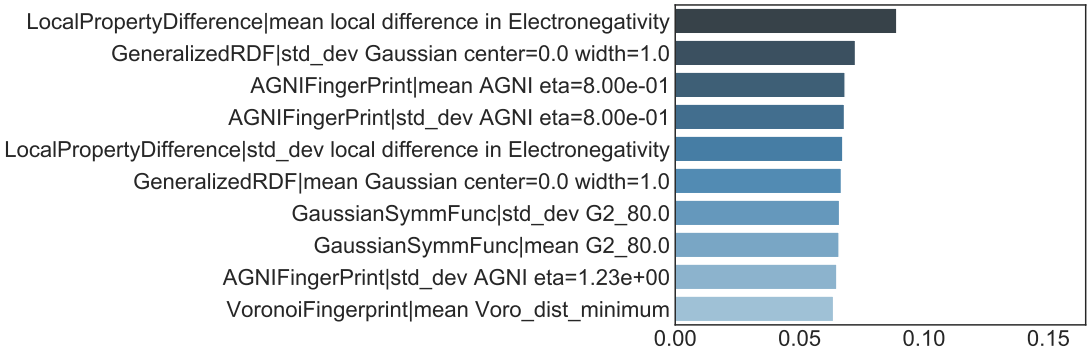

Figure 4.5.9: 10 highest ranked *site* features according to their NMI with the formation energy.

related to compositional features. This conclusion is important and should be taken into account when selecting features for the final predictive model.

4.5.3 Final comments and conclusion

Let us conclude this section with some final comments on why the preceding results are of high interest.

Convergence

The convergence heatmaps offer much more than guaranteeing the correctness of the results. They show us roughly what size is needed to find relations between features and between features and targets, which is of high interest when dealing with limited datasets. For example, most maps converged around 1000 samples which gives some rough minimum threshold to perform machine learning. Secondly, it also enables one to detect sparse features (at least for the considered dataset), which are prone to overfitting and should thus be removed.

NMI-grams

The resulting NMI-grams enables us to find relations between features, detect closely related features (such as the Coulomb and sine-Coulomb matrix) or complementary features. Moreover, it gives a broader understanding of the features by showing their relations with targets or other features. Finally, they will pave the way to the upcoming selection algorithm.

4.6 Selection algorithm

The next chapter will deal with predicting vibrational-thermodynamical properties of materials (such as the vibrational entropy and enthalpy), based on the previous presented feature set. All combined, there are more than 2000 features for approximately 1400 sampled materials. When the dimensionality of the feature space is high, and especially compared to the training set size, a strange behavior can be expected. This is mainly referred as the curse of dimensionality in the litterature [86], [90]. When the input space increases, some very surprising facts occur such as the concentration of the Euclidean norm and distances, which is not desirable as most algorithms rely on these properties. Moreover, a high dimensional input space means more "void" around the samples, and one is never sure to interpolate correctly, giving rise to overfitting [91]–[93]. All this together could lead to a significant drop in generalization performance of the predictor. Therefore dimensionality reduction of the input space is often a good idea.

Furthermore, reducing the number of input features decreases the time complexity of the algorithm, making the training *and* prediction phase faster.

Last but not least, when one reduces the input space by selecting a subset of features, the model becomes often more simple to interpret in terms of dependent physical variables. This is especially true compared to more advanced feature extraction models that create a combination of various, often incommensurate, features which makes physical interpretation very often impossible.

In short, feature selection will be performed for the following three reasons:

1. Avoid the curse of dimensionality.

2. Speed up training and predicting time.
3. Physical interpretation of the input space.

There exist various approaches to select a subset from a larger set of features [50], [84], [94], [95]. They can be divided in three groups: filters, wrapper and embedded methods, based on their selection criterion. Filters give a score to each feature based on a statistical measure, that is often univariate, such as the correlation. Features are selected in the order of their score, and an additional stopping criterion such as the total number of features or minimum correlation is needed. In contrast, wrappers select features based on an iterative procedure with respect to the performance of a given model. Features are progressively added (or removed) following for example a greedy search or a genetic algorithm and the corresponding prediction error is computed. Features giving the lowest validation error are kept iteratively until no further improvement is found. The advantages of wrapper are that they are model-aware and have an easy criterion of relevance (validation error), but are however often very slow as the model should be fitted at each iteration. Filters on the other hand are simpler and faster but it is more difficult to define a stopping criterion and they are moreover not model-aware. Finally, embedded methods are models that have an embedded feature selection algorithm such as regularization for example, which basically penalize the use of features.

Results from the previous section contain a vast amount of information on the features themselves and on their common relations, that must be exploited to choose the features wisely. Intuitively, one would choose features that have a high relation, i.e NMI, with the target output such as listed in figure 4.5.1 , but removing the ones that are strongly related as they increase the input dimensions without much additional information. In other words features are chosen in order to maximize relevance (feature-target correspondence) and minimizing redundancy (feature-feature correspondence). Because manually selecting features is nor desirable nor feasible, a simple algorithm is proposed here based on this relevance-redundancy criterion (named RR-algorithm).

Given a set of features F with k features, the $k+1$ feature f_x is selected as having the highest score RR,

$$RR(f_x) = \frac{\mathrm{NMI}(f_x, y)}{\left[\max_{f_i \in F}\left(\mathrm{NMI}(f_i, f_x)\right)\right]^p + c} \tag{4.5}$$

where $\mathrm{NMI}(\cdot)$ is the normalized mutual information and p,c two parameters to be determined. When $k = 1$, i.e for the first feature, the feature with the highest NMI with the target is chosen. This score is proportional to the NMI with the target output, but additionally penalizes features that have a high relation with any feature in the existing feature set, by taking the maximum of the NMI in the current feature set. Parameters p and c describe the importance given to the relevance-redundancy ratio. Increasing c will neglect the redundancy and only rank the features according to their relevance, while changing p differentiates more or less features with a similar redundancy. It is taken as assumption that both p and c are not target nor model dependent, which is a quite strong hypothesis, but is taken for simplicity. This means that these parameters are basically some ratio of relevance-redundancy that works well for all problems, but are not known *a priori* and thus need to be fitted once and for all.

In this regard, this criterion acts as a simple filter. However, as stopping criterion, a wrapper approach will be followed, by increasing the number of features until no significant decrease in validation error is found. This algorithm is schematically represented in figure 4.6.1.

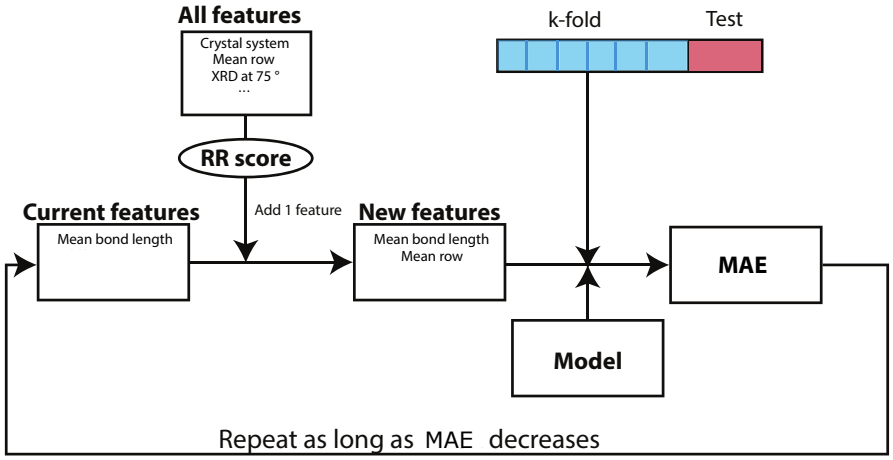

Figure 4.6.1: Schematic representation of the RR-algorithm. Features are iteratively added based on the highest RR-score, followed by a cross-validation. As long as the mean absolute error (MAE) decreases, the procedure is repeated.

4.6.1 p AND c DETERMINATION

We will now try to optimize the previous selection algorithm towards the relevance-redundancy ratio, i.e selecting values for the parameters p and c. A neural network, based on transfer learning, will eventually be used as final model, as will be discussed in the next chapter. With this in mind, we will study the performance of a simple one layer neural network, with respect to the number of features and the relevance-redundancy parameters. This will enable us to fix the values of p and c for the remaining work. Remember, that it was assumed that both p and c are model and target independent, which means that any target or model could be used. However, choosing a model and target close to the final one can only be beneficial, and therefore a one layer neural network is chosen, with as target the vibrational entropy at 300K. This is close to what will be used later, but fast enough to train, enabling us the make a grid-search on parameters p and c in a reasonable time.

METHODOLOGY

For the model, a very simple neural network is used, containing only one hidden layer above the input and output layer. This hidden layers contains as many neurons (units) as the mean number of input and output neurons, with a bias of 16 neurons, i.e

$$N_h = \frac{1 + N_{\text{input}}}{2} + 16 \tag{4.6}$$

with N_h the number of hidden units and N_{input} the number of input units, which corresponds to the number of features. The addition of 16 neurons ensures that when only a few input

features are used enough hidden units are present to map the complex relation. A rectifying linear unit function is used as activation function for the hidden layer, while a linear activation function is chosen for the output layer. The **keras** framework with a **tensorflow** backend is used for the actual implementation [96].

As target, the vibrational entropy at 300K, per atom is used, as published in Ref. [11], a high-throughput density-functional perturbation theory for inorganic materials, containing 1245 entries.

In order to study the impact of the RR-algorithm, the number of input features is progressively increased following the selection score, for different values of p and c. For each set of features the previous model is trained and validated following a 5-fold cross validation. The mean absolute error is used as validation metric, and is kept as final score for a particular combination of p, c and the number of input features.

Results

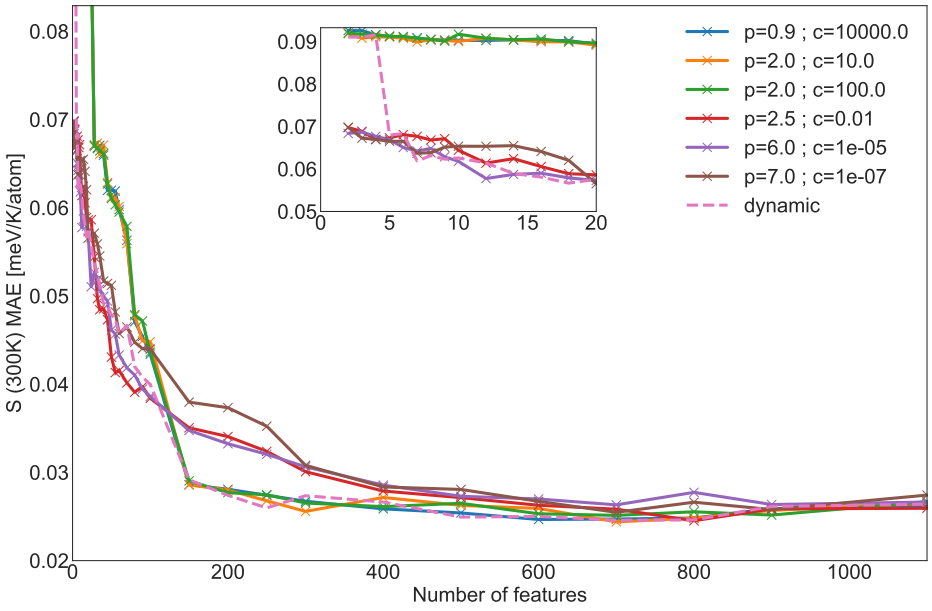

Figure 4.6.2: Validation mean absolute error (MAE) in function of the number of features for different values of (p, c) (as defined in Eq. (4.5)).

Figure 4.6.2 depicts the mean absolute validation error in terms of the number of input features, for different combinations of p and c. First it is seen that the validation error decreases with the number of features, and this for all combinations of parameters. Therefore, no overfitting is found as was expected initially, which could make this study look unnecessary at first. However, it can be seen that beyond 200 features no strong improvement is found, if choosing the right set (p, c), which is a motivation on its own to reduce the input space. Moreover, the model that was used in this study is pretty simple with only one hidden layer a few hidden neurons (less then 500), which makes it difficult to achieve overfitting. Therefore, with more complex models, as will be the case in the next chapter, an increase in the MAE with the number of features can truly be expected. From this, it makes thus

sense to limit the input space, even if it increases (only slightly) the validation error. One could for example choose the green curve, and stop at 200 features, which gives results of roughly the same accuracy then other curves at much higher number of features, but a slightly different approach will be chosen as described next.

When looking at the effect of the redundancy-relevance ratio, the curves can be split in two groups. When having a high value for p (around 2 to 10) and a low value for c (around 0 to 0.1), which corresponds to a relative strong importance to redundancy, the MAE drops quickly for the first features, as can be seen from the inset (especially already from the first 2 features). This drop in MAE is significant until 80 features, and much less beyond. In contrast, low values for p but high values for c, which more or less ignores redundancy, give rise to a significant decrease much further, around 100 features. These two family of curves are clearly distinguished when looking around 100 features on the previous figure. This behaviour is interesting and would mean that when selecting only a few features, it is best to choose them not too redundant. Intuitively, we want features that cover a broad range of different properties. However, when considering larger feature sets (around 100) it is better to give more importance to relevance, in order to capture the smallest relations with the output, without bothering on redundancy as we can afford to have many features. The problem with the first family of curves (giving much importance to redundancy), is that from around 100 features it penalizes itself by forcing to choose new features that are not redundant with the current set, but because this set is quite big at that point, only rare (often not useful) features are chosen, which explains why the two families of curves cross each other around 120 features.

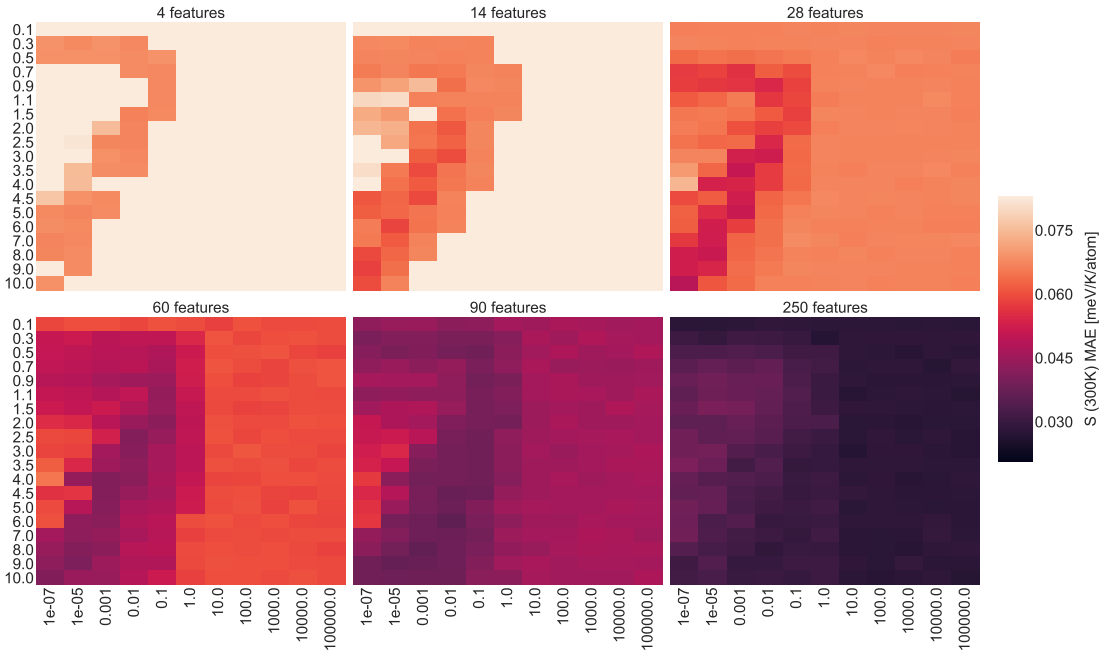

Figure 4.6.3: Validation mean absolute error (MAE) for different sets of features chosen according to the RR-algorithm, stopped at a fixed number of features. It is seen how the minimum in MAE shifts from left under to the right, meaning that redundancy is only important for the first few features.

Therefore, a good compromise would be to give a high importance to redundancy at the beginning (first few features), and then gradually decrease this importance. In other

1	Mean average bond length (**AverageBondLength**)
2	Mean row in the periodic table (**ElementProperty**)
3	GRDF centered at 3.0 Å and width of 1.0 Å(**GeneralizedRDF**)
4	HOMO element (**AtomicOrbitals**)
5	Crystal system (**GlobalSymmetryFeatures**)
6	Mean minimum vornoi volume (**VoronoiFingerprint**)
7	Mean square co-planar CN 4 (**OPSiteFingerprint**)
8	Mean A:2 (i.e angular) environment (**ChemEnvSiteFingerprint**)
9	Mean spherical harmonic l=7 (**BondOrientationParameter**)
10	Standard deviation of the Voronoi volume(**VoronoiFingerprint**)

Table 4.6.1: The first 10 features chosen by the RR-algorithm, with the corresponding class in parentheses (see chapter3).

words p and c should vary dynamically as a function of the number of features, going from the first family of curves to the second. This idea is confirmed by the heatmaps in figure 4.6.3, depicting the MAE for a much larger grid of (p, c), at different stages of the feature set size. It is clearly seen how the minimum shifts from redundant-import zones to redundant-unimportant zones, when increasing the feature set.

In consequence, it is chosen to dynamically change the values of (p, c) as follows:

$$\begin{cases} p = 6 - 2\log(n+1) & \text{if} \quad p > 0.1 \\ p = 0.1 & \text{otherwise} \\ c = 10^{-4}\exp\left(\frac{n}{10}\right) \end{cases}, \tag{4.7}$$

where n is the number of features. For our limited experience, this approach appears to work well and will be used in the upcoming work. The corresponding performance is drawn as a broken line in figure 4.6.2. Alternatively fixing p to 2 and changing c such that $c = 10^{-5}\ n^3$ works also well and is probably more simple and satisfying.

4.6.2 Feature selection for different targets

Now that the parameters of the selection algorithm have been decided, one could be interested in the nature of the selected features. Table 4.6.1 contains the first 10 selected features for the RR-algorithm with the vibrational entropy at 300K as target. It shows that a diverse set of features is chosen from the composition, structural and environment related modules, and are globally related with the bond length.

What if one would want to predict the entropy at another temperature? Are the chosen features roughly the same or not at all? To answer this question, the feature set is computed for the vibrational entropy for various temperatures and compared. Because a manual comparison is not feasible, the Jaccardian similarity between two sets will be used. Given a set $\mathcal{S}_1$ and a second set $\mathcal{S}\in_2$, the Jaccardian similarity between these two sets is defined as,

$$\text{Jacc}(\mathcal{S}_1, \mathcal{S}_2) = \frac{\|\mathcal{S}_1 \cap \mathcal{S}_2\|}{\|\mathcal{S}_1 \cup \mathcal{S}_2\|} \quad \in [0, 1], \tag{4.8}$$

with higher values indicating a higher similarity.

Results are depicted in figure 4.6.4 for the different temperatures, and were chosen such that some are close together (e.g. 300K and 310K) and other farther apart (e.g 100K and 500K). Two main observations are made: first, increasing the number of features increases the overall similarity. Second, the closer the temperature, the closer the similarity. For example the entropy at 300 and 310K have already a similarity of 80 % for 10 features, and increases up to 90% for 500 features. However, the similarity between the entropy at 100K and 500K is 25 % at 10 features, but increases rapidly by being above 50% at 100 features. Note that a slight difference in features at a low set size number is quite important as the first features are commonly the most important.

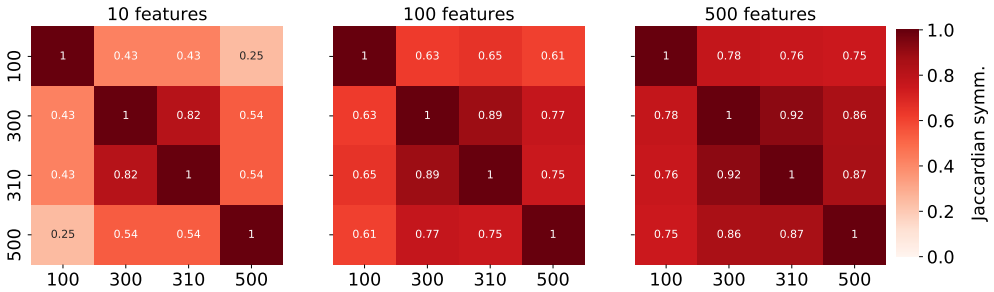

Figure 4.6.4: Jaccardian similarity between feature sets chosen by the RR-algorithm at different temperatures. The similarity was computed at three feature-set size: 10, 100 and 500 features.

In conclusion, we can state that even if we consider the same property at different temperatures, a non negligible difference exists albeit not extreme. These differences will be even more noticeable when considering other properties, for example the specific heat at 300K compared to the vibrationnal entropy at 300K. It is thus important to recompute the feature set for different properties even if they seem similar.

In particular, in the next chapter various thermodynamic properties will be predicted from the same features set. From what is discussed here, it will be chosen to compute the feature set for each property at an interval of 50 Kelvin and take the union over all resultant features.

4.6.3 Comparison with other feature selection methods

To close off this section, the presented RR-algorithm will be compared with a few other selection algorithms.

Other algorithms

The RR-algorithm is compared with two other selection algorithms and two extraction algorithms presented hereafter.

Correlation. A simple selection algorithm based on the Pearson correlation, where the features are selected in order of their highest correlation with the target variable.

MI. Features are selected solely based on their mutual information with the target variable, and thus neglecting redundancy. Features with the highest MI are chosen first.

PCA. The principal component analysis (PCA) projects the input variables on a new set of axis that maximize the variance along each axis. It thus diagonalizes the covariance matrix and is thus a linear transformation (can be seen as a rotation in space). Each axis has a score corresponding to the explained variance, and are chosen in decreasing order.

PLS. Also known as PLS1. The partial least square (PLS) decomposition, finds iteratively a new axis that is a linear combinations of the features such that the covariance between this new component and the target variable is maximized:

$$Cov(\mathbf{X}\alpha, \mathbf{y}) \propto Corr(\mathbf{X}\alpha, \mathbf{y}) \cdot \sqrt{Var(\mathbf{X}\alpha)} \tag{4.9}$$

where α is the new linear combination giving the first component. A second, third, etc. component can be found by repeating the procedure with the constraint of being uncorrelated with all previous components.

From Eq. (4.9), it appears as a mixture between a linear regression (correlation factor) and PCA (variance factor).

Results

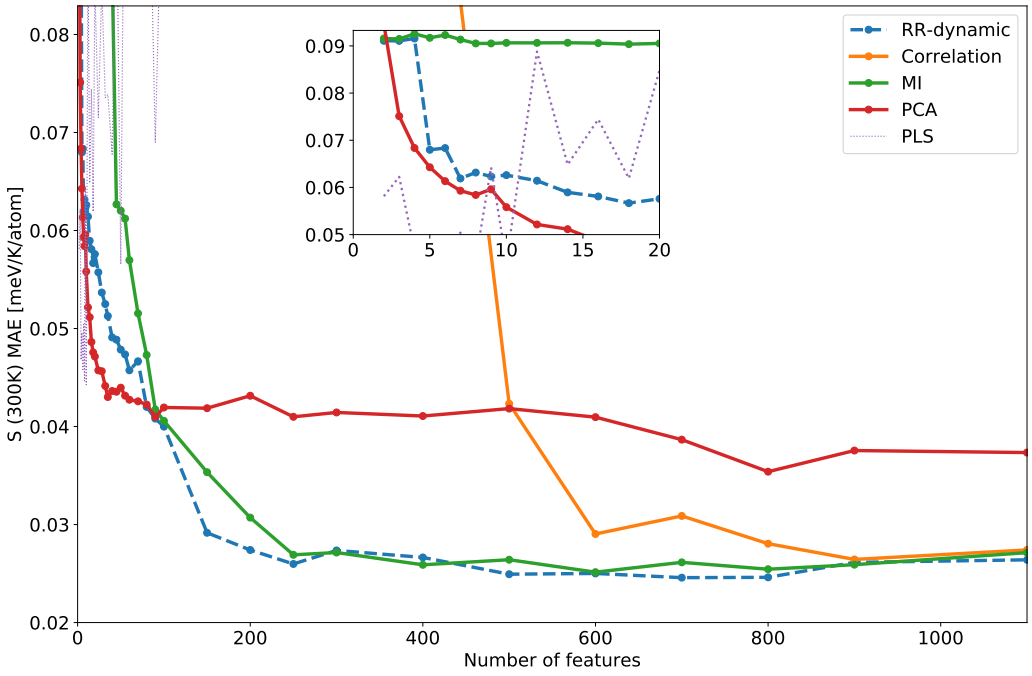

Figure 4.6.5: Comparison in the ability to select wisely features for our neural network following different approaches. The validation mean absolute error (MAE) in function of the number of features (components) is depicted for the different algorithms presented in the text.

Figure 4.6.5 represents the mean absolute error (MAE) as a function of the number of features (components) kept for the different selection (extraction) algorithms. The same model and target variable as explained in section 4.6.1 is used.

Two algorithms are performing very poorly, namely the correlation and PLS algorithm. Concerning the correlation, this is due to the linear nature of this measure which is not adequate for the considered target. Concerning PLS, it performs poorly on the validation set, but well on the training set (not shown), which indicates a clear sign of overfitting. This can be explained by the nature of the transformation: a linear combination is found on many features (approx. 2,000) which happens to be correlated with the (training) target (which is always possible given the high amount of targets compared to the 996 training targets). However, this will only be a spurious correlation, enhanced by the neural network, that will not perform well on the validation set. This overfitting is naturally enhanced by increasing the number of components as is seen from the figure.

Interestingly the PCA is performing the best for a small number of features, as it is seen in the inset, but less accurate when dealing with a higher number of features. This is because it can capture most of the input in a few components (the extraction principle), but this becomes less useful and more confusing for the neural network at high number of features.

Finally, the MI-algorithm has the expected behaviour as it corresponds to the RR-algorithm with $c \to \infty$, namely very poor performance at low number of features (see inset), but joins the dynamic-RR at higher number of features.

As a conclusion, it is seen how well the dynamic RR-algorithm performs. At low number of features it is only slightly worse then a PCA extraction, but has the advantage of being interpretable. At higher number of features it shows the lowest validation error on the considered algorithms.

4.7 Conclusion

This chapter was dense in information, but enabled us to better understand the features by computing their common relations, as well as to build a smart selection algorithm based on a relevance-redundancy criterion.

This chapter started by presenting a criterion of relevance and redundancy based on the concept of normalized mutual information, to go beyond the limitation of the correlation. From this, various NMI-grams were constructed to better understand the redundancy and relations between the features. For example, it was seen how the elemental features are globally very related together. Many obvious relations were found but also less-obvious relations such as the importance of oxygen on many properties. Concerning the structure, a clear redundancy was found between the Coulomb and sine-Coulomb matrices. These latter representations were also shown to be both structural and composition related. Furthermore the relation between the spacegroup and crystal system was fully retrieved. Concerning the site-features two groups of features were distinguished: one based on local motifs and a second one based on the radial distribution function. These are only a few examples, and many more were given in this chapter. Finally, concerning the vibrational entropy, it was seen that the structure was a very important factor. For instance, the mean bond length, mean row, atomic radius, Coulomb matrix and RDF-based features belong to the features with the highest NMI for this target. In contrast, it was seen that the formation energy is related to very different features such as the electronegativity.

Moreover, much attention was put on the convergence of the NMI-grams. The convergence

is not only important for the sake of correctness but it also shows how many samples are needed in order to detect patterns. It also enables us to find sparse features, by detecting the presented category three points. For example, most element-fractions, bond fractions, f-orbital filling and exotic local environments were identified as rare. As they are prone to overfitting, they will be discarded in the upcoming work.

This first part enabled one to better understand how the different features are inter-related and offered new insights on determining properties for the vibrational entropy. Next to this qualitative analysis, it also offered a quantitative basis for the second part of this chapter, namely a smart feature selection algorithm. It is based on an iterative process were a relevance-redundancy score, presented in Eq. (4.5), is computed for each feature in order to rank them. This algorithm was shown to be performing well, even when compared to well-known extraction algorithms. It will form the basis of the next chapter, where a model will be built on top of the best selected features.

Chapter 5

Thermal Transfer Neural Network

5.1 Introduction

Vibrational properties are of key importance to understand physical phenomena such as thermal conductivity, superconductivity, and ferroelectricity among others. Experimental phonon spectra are only available for a very limited number of compounds. Therefore, high-throughput methods based on density functional perturbation theory (DFPT) in the harmonic approximation have been developed (see Ref. [11]), giving rise to an important amount of data. However, these computation are extremely time consuming and therefore very limited in their amount of materials. Clearly, there is a lack of quick screening methods for thermodynamic data, which could be solved by machine learning.

This chapter will therefore close off this work by proposing a new approach that enables one to predict vibrational properties of crystaline solids, by the so-called thermal transfer neural network (TTNN). We have paved the way to this task, by generating, analyzing and selecting our features in previous chapters. As a final step, a neural network based on transfer learning will be presented, taking full advantage of the knowledge stored in the dataset.

The first upcoming section will briefly describe this dataset in terms of material distribution. This is followed by a discussion on model complexity and training size as these are very important concepts when building a model. The third section will discuss the actual implementation of the TTNN in more details. Finally, the enumeration of some future improvement will close off this chapter.

5.1.1 Code availability

The code that was used to obtain the results (including the graphical illustrations) of this chapter can be found on the *github* repository of this work, see Ref. [89].

5.2 Dataset presentation

Learning will be performed on the previously mentioned phonon dataset [72], containing vibrational data for 1,245 inorganic compounds, computed in the harmonic approximation based on DFPT. It contains the vibrational entropy, the Helmoltz free energy, the vibrational contribution of the internal energy and the heat capacity from 5 to 800 Kelvin in steps of 5 Kelvin. Typical values for the different thermodynamic quantities are presented in table 5.5.2, second column, in meV/atom or meV/K/atom.

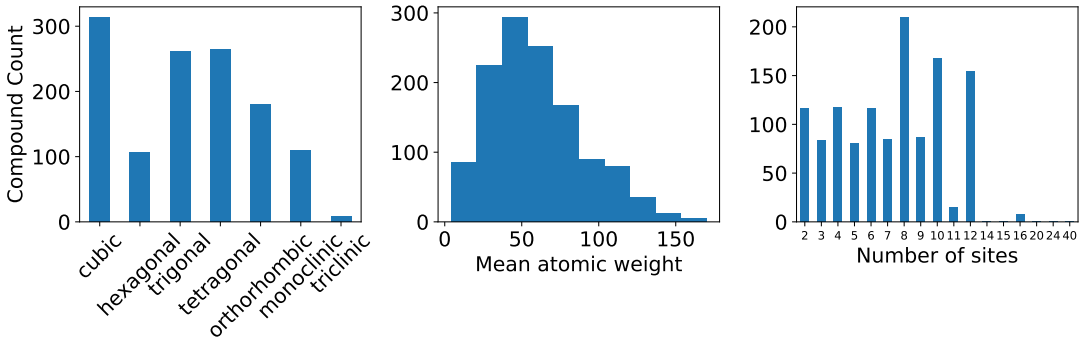

Figure 5.2.1: Various histograms showing the distribution of the samples used for predicting the vibrational properties. It contains a wide variety of elements and crystal structures.

The dataset contains various crystal compounds, ranging from simple mono-elemental compounds to complex semiconductors. These 1,245 materials cover the 7 symmetry groups, 84 space groups and 64 elements. Most compounds are ternary alloys and there are no materials with more than 5 different elements. As can be seen from figure 5.2.1, the mean atomic mass ranges widely, confirming the variety of elements present in the dataset. Moreover, from the same figure, it is seen that the number of sites in the primitive cell ranges from 2 to 40, with most primitive cells having less than 10 atoms. Finally, for the interested reader, appendix B contains supplementary histograms about the data distribution.

One could wonder how the target variables are related to the input features. Moreover, it is certainly interesting to see if the input space (i.e. material features) could be partitioned into similar regions by using unsupervised learning algorithms. To answer these questions, the dataset will be graphically represented by using dimensionality reduction algorithms, see figures B.0.3 and 5.2.3. In particular, three dimensionality reduction algorithms will be used: PCA, PLS and the previously presented RR-algorithm.

The PCA algorithm finds a rotation of the input space such that the variance along each axis is maximized. It is thus an unsupervised learning algorithm as it uses the input space only. Here, it is used as a visualization tool by limiting the number of dimensions to three. The result is represented in figure B.0.3, where the target variable is additionally depicted by usage of a color attribute. The target values are the vibrational entropy at 300K. It is seen how the first axis mainly splits the data into two subgroups. When investigating these subgroups, it is seen that the smaller one mainly contains 'rarer' compounds with f-electrons or rare local environments. Moreover, from the 3D PCA projection, it is possible to see how the target variable is not totally randomly distributed, which shows that the relation is partially grasped by the principal axes.

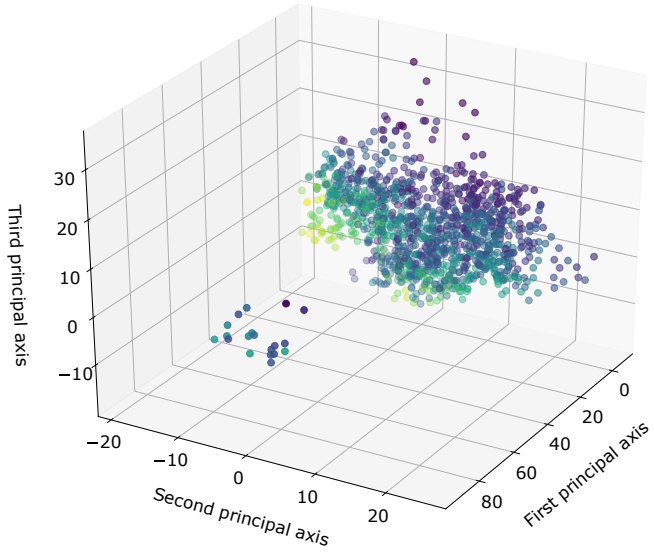

Figure 5.2.2: Dimensionality reduction of the input space by means of PCA. The color attributes represent the value of the vibrational entropy at 300K. The projection isolates 'rare' compounds into a second subgroup.

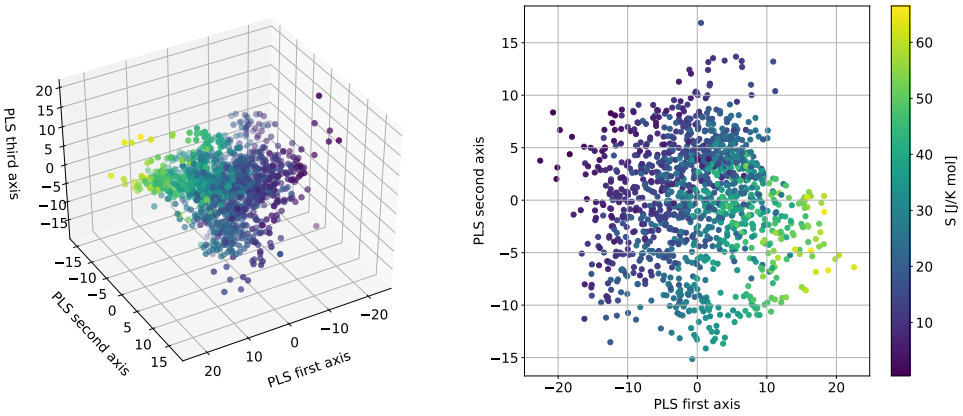

Figure 5.2.3: Dimensionality reduction of the input space by means of PLS. The color attributes represent the value of the vibrational entropy at 300K. The PLS algorithm is capable of rearranging the input space based on the vibrational entropy using a pure linear transformation. The right image shows how the first two features are clearly correlated with the target.

The PLS algorithm will maximize the covariance between the components and the target variable. It is not unsupervised as it uses the target value, and can therefore be seen as a linear regression that additionally tries to maximize variance along the chosen axes. The results are shown in figure 5.2.3. It is quite remarkable to see how the vibrational entropy is well partitioned using this linear model. Particularly, the first two components,

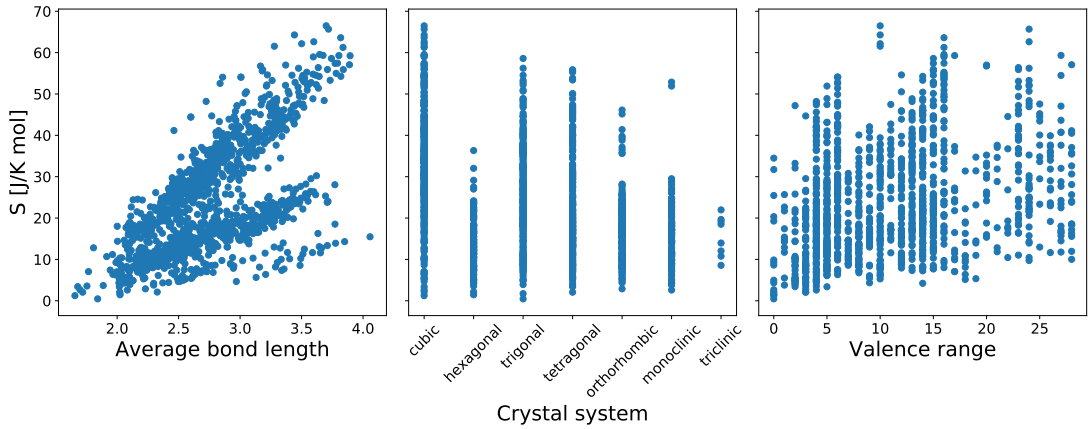

Figure 5.2.4: Univariate visualization of the first three features selected by the RR-algorithm for the vibrational entropy at 300K.

i.e. subfigure (c), capture well the relation by assigning high entropies with a high value for the first component and a low value for the second component. In short, this shows that a linear model can definitely be used for a fast and rough estimation, although certainly not having the best accuracy.

Finally, to illustrate the previously presented RR-algorithm that will be used as feature selector, a univariate visualization is done for the first three features in figure 5.2.4 . One can see how the average bond length is a good indication for the vibrational entropy at 300K. The valence range certainly reduces the uncertainty but is less indicative then the mean bond length, taken alone. The same happens for the crystal system. However these features are very complementary (as per the RR-algorithm), and by combining them together, new patterns could be discovered. For example, note how the mean bond length univariate plot has three main branches that behave more or less linearly. By restricting this graph to cubic systems, it is seen that principally the middle branch remains. This shows how the crystal system is in fact very indicative as opposed as it appeared first.

5.3 On training size, model complexity and the bias-variance tradeoff

According to the 'No Free Lunch Theorems' [97], no ML algorithm is universally superior. This means that one should consider a case by case approach when building models. Many different algorithms exist based on different methods and of different complexities. Some of them were presented in chapter 1. When choosing and/or constructing a model, it is very important to confront its complexity to the dataset size. A very common way to estimate the complexity of a model is to count the number of parameters (i.e weights). Simpler models such as linear regression, LASSO or ridge regression will typically contain a few tens to a hundred of parameters to fit, while deep convolutional neural networks typically contain some ten thousand of parameters if not more. This model complexity is of great importance when considering the test error, i.e. the ability of a model to generalize to new data. In this regard, figure 5.3.1 depicts the bias-variance trade-off. Very simple algorithms (or when very few features are present) will tend to have difficulties in learning non-trivial relations, and tend to have a biased test error, also known as underfitting. On the other

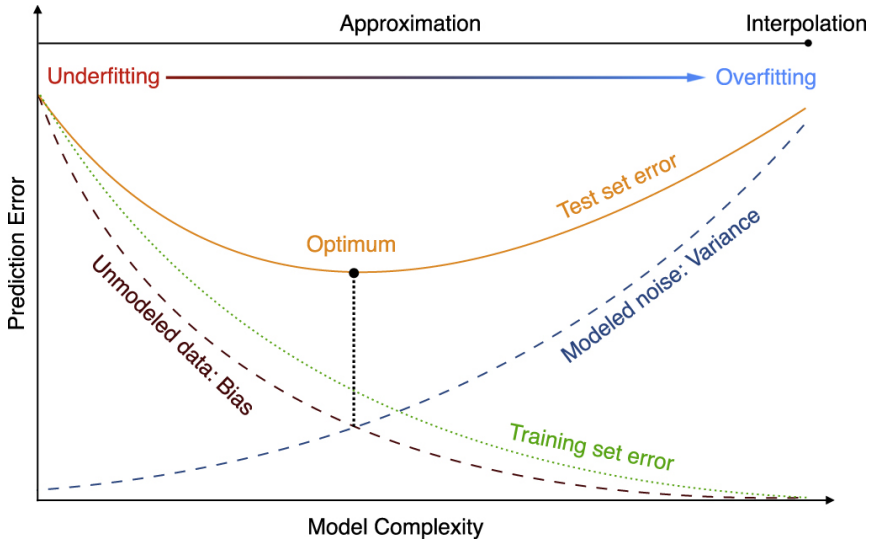

Figure 5.3.1: Bias-variance tradeoff. Depending on the model complexity and training size a trade-off between bias and variance is found on the prediction error given by the test set.

hand, very complex models with many input features will capture every single detail in the input data, noise included, which results in a very low training error. However, they will generalize very poorly and therefore suffer from a high variance, also known as overfitting. These are the two extreme cases depicted in figure 5.3.1, and ideally one wants to be in the middle of this trade-off, close to the optimal test error. In practice, this means that the number of parameters should not exceed too much the number of samples in the training set, or otherwise the model could just be a databank remembering the samples.

In the literature, machine learning is commonly performed starting from 500 to 1,000 samples, while more intensive deep learning is done from 10,000 samples. In practise many interesting problems, including the present one, are very limited by the amount of available data. Intrinsically, hard problems in the field of material science that could benefit from machine learning suffer from this very problem. This is a major limitation to many of the fancy techniques presented in chapter 2 based on deep learning, that drive much attention nowadays, including the CGCNN, which is disappointing given the performance that these algorithms could have when properly trained. Therefore, this chapter will try to propose a solution to this problem, by combining effective, more complex, (deep) models with relatively small datasets, by using what is known as transfer learning.

5.4 Transfer Learning

Traditional machine learning will build a new predictive model on a per task basis, and repeat his independently on every new task. Transfer learning however, is a very general technique that uses the knowledge gained from a first problem to a second problem. In other words, previously learned knowledge from a first task is *transferred* to a second one. This is schematically represented in figure 5.4.1. Of course, this requires that both tasks are more or less related, otherwise no benefit will be found.

It is commonly used in the field of image recognition, see Refs. [98]–[104], where deep neural nets are partially reused for a new task. For example, a convolutional neural network is first trained on recognizing cars. Then, in order to recognize trucks, the first convolutional layers

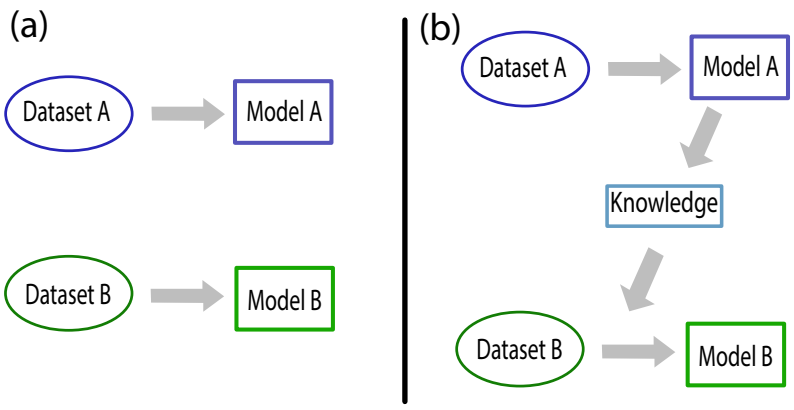

Figure 5.4.1: Traditional machine learning (a) versus transfer learning (b). Transfer learning reuses knowledge gained from a first problem to improve accuracy on a second problem.

(and sometimes also the first fully connected layers) are reused with the previously learned weights, and appended with some new randomly initialized fully connected layers. Finally these latter new layers are optimized towards the new task while the first ones are usually frozen or fine tuned (i.e. small learning rate). This works well, since many recognition task have the same common first convolutional layers (i.e Gabor-like filters), while latter layers are more task specific [102].

Transfer learning has two main advantages. First, it speeds up the learning process. By reusing already tuned layers, only a few epochs are needed in order to find the new local optimum. Secondly, and mainly why it will be used here, transfer learning enables one to learn the second task with a limited smaller dataset, under the condition that the reused layers have been properly learned on a bigger dataset with the first task. Typically, in the image recognition field, when a new recognition task is needed to be performed (e.g. scene recognition) on a limited dataset, previous neural nets trained on much larger datasets are used as a starting point, with a resulting increase in performance.

This idea will be carried over by applying it on the material science field. It is reasonable to think that if a multilayer fully connected neural network would be used to predict a certain property, the first few layers would contain some general representation of the material itself. One could imagine that the first layer is the genome of the material, well understood by the machine, while latter layers will decode this genome to the final property.

This first layer, or more generally the first block, would be the same for every single task, while subsequent layers are task specific. This leads us to a very simple two block model, where a first bunch of layers create the materials genome while a second block of layers decode the genome. The first block is sequentially reused and fine tuned for every new task (and thus getting better and better) while the second block is target specific. Moreover, the complexity of this first block could be quite high (several layers or many neurons), as it is passed, or learned on many tasks (thus 'seeing' many samples), while the second block would be rather simple. This scheme would naturally result in a better generalization performance. See figure 5.4.2 for a schematic representation.

In order to test this sequential idea (and thus to confirm the added-value of transfer learning for materials), a simple preliminary experiment is executed. An 'encoder' block is built

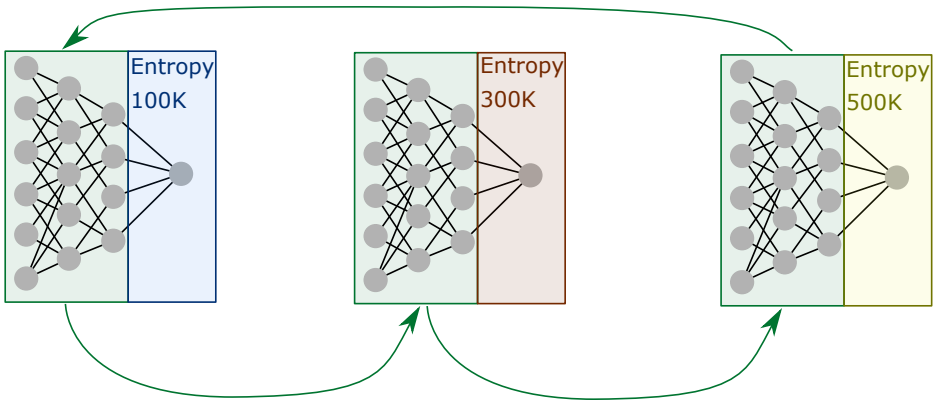

Figure 5.4.2: A two block model for material property prediction based on a neural network. A first block (in green) of layers encodes the features into a materials genome, a well understood representation by the neural network. A second block of layers decode the genome into various material properties, at various temperatures, represented in blue, red and yellow. The same green encoder is used for each task, and sequentially fine-tuned from task to task (green arrows), while decoders are task specific.

consisting of two hidden layers of size 64 an 32, while 80 'decoders' - consisting of one hidden layer of size 32 - are built too, in order to decode the genome to the entropy at 80 different temperatures. They are then sequentially trained following the previous scheme. The MAE is depicted in figure 5.4.3 for the vibrational entropy at 105K, with a different background color after each transfer of the encoder to the other 79 models before coming back. One can seen how the validation error decreases slightly but significantly (after a short perturbation), especially from the green to the red zone, while the training error does not change much. This simple test shows us that the network is indeed learning from the other properties, and that transfer learning can be successfully applied to the material science field. Note that the learning rate in this experiment was probably too high, but no additional effort will be put in this sequential model as it is globally rather slow to learn (many small models). An equivalent and more efficient solution exists and will be discussed next.

Now, instead of fine tuning the 'genome encoder' sequentially, it would be equivalent and even more efficient to learn the different tasks jointly. It has the advantage of optimizing the 'genome encoder' on all tasks in parallel, instead of forgetting partially previous tasks. This method of jointly learning is in fact one special case of transfer learning, see figure 5.4.4 [104].

Going even further, one could imagine that next to a general material encoder, a second block could transform this representation to a more specific thermodynamic representation that is shared commonly for many third level predictor blocks, predicting different thermodynamic properties. Another second level block could be build, but now shared by mechanical-property third level predictors. A fourth level could be added that splits different predictors based on the actual property, but sharing different temperature predictors. This would *in fine* lead to a hierarchical tree-like network architecture, as described in figure 5.4.5. This idea forms the basis of the upcoming Thermal Transfer Neural Network (TTNN).

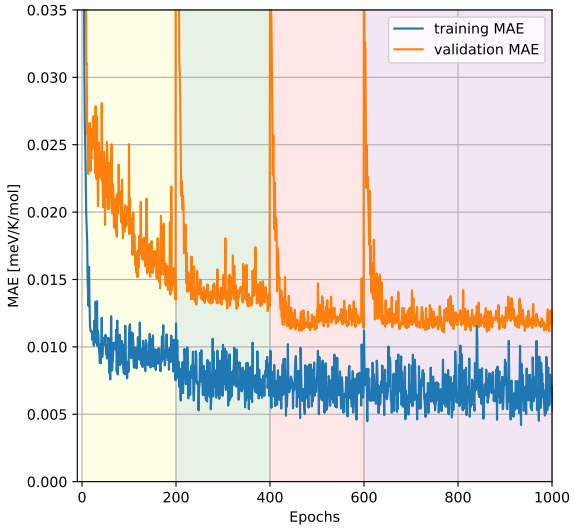

Figure 5.4.3: Empirical test of transfer learning on material properties. An 'encoder' is sequentially trained on 80 vibrational entropies of different temperatures, while each property has its own 'decoder'. The MAE of the vibrational entropy at 105K is depicted here, where a change in background color indicates that the encoder has been taken off and learned on the 79 other properties before coming back.

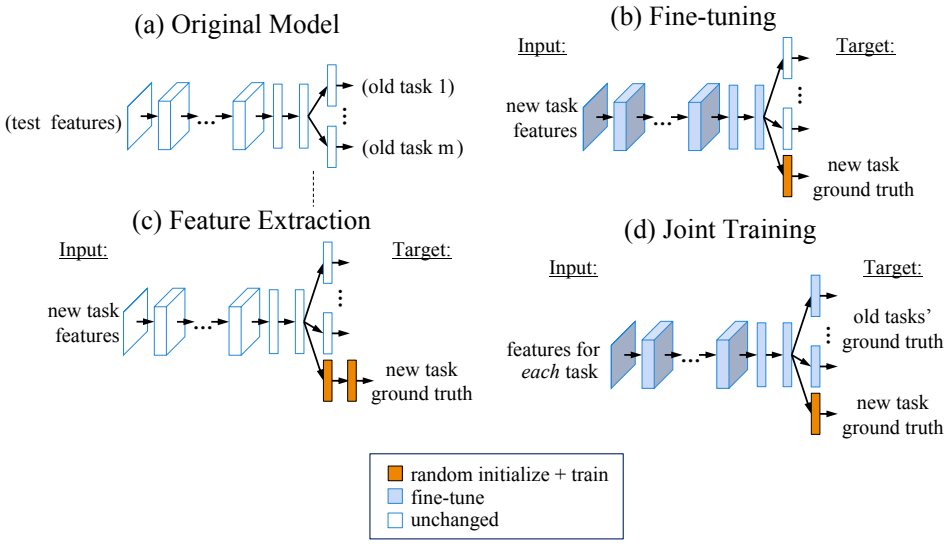

Figure 5.4.4: Schematic representation of the different forms of transfer learning. New layers are added to the original model (a) for a new task. These new layers are trained while the original layers are either fine-tuned (b), left unchanged (c), or jointly trained with the new layers (d). Adapted from Ref. [104].

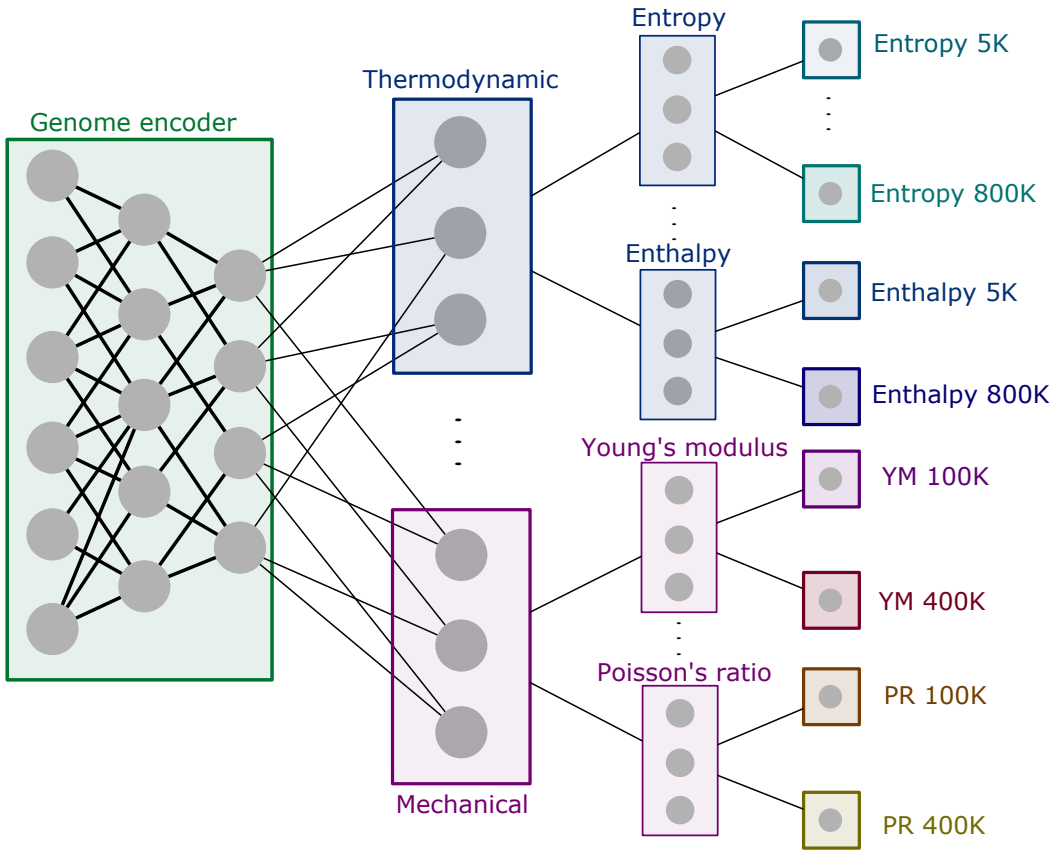

Figure 5.4.5: A hierarchical tree-like neural network for material property prediction. The first green block encodes a material in an appropriate all-round vector, while subsequent blocks decode and re-encode this representation in a more target specific nature. First level blocks are visited (i.e. optimised) by many samples, and therefore more complex, while latter blocks are less visited, and therefore less complex, which increases the overall generalization. Simple black lines are used for representing the many connection between neurons.

5.5 Thermal Transfer Neural Network (TTNN)

5.5.1 Overview

We will now combine both novel ideas presented in this work, i.e the RR-algorithm and the tree-like transfer neural network, and apply it on the vibrational thermal data which will form the Thermal Transfer Neural Newtork (TTNN).

The input will consists in a set of wisely chosen features, by the RR-algorithm, and will be explained in the upcoming section called *Feature selection*. The output covers the four vibrational thermodynamic properties (i.e. Helmoltz free energy, vibrational entropy, vibrational contribution to the internal energy and specific heat) from 5K to 800K in steps of 20 K. This results in 160 targets coming out of the TTNN. The detailed architecture will be presented in the section called *Architecture* while the model prediction accuracy will be discussed in the section called *Performance assessment*.

5.5.2 Feature selection

The first step in building our final model is to select the features that will be fed to the neural network.

From the results in chapter 4, the RR-selection algorithm will be used (Eq. (4.5)) together with the optimal parameters (Eq. (4.7)). There is however a slight additional subtlety to the problem. Indeed, the proposed model is a multi-target model, predicting four properties over a range of temperatures, while the RR-algorithm is a single target algorithm. It was found that properties close in temperature have a very similar optimal feature set, while more distant temperatures have a dissimilar optimal feature set, see figure 4.6.4, Chapter 4. Therefore, it was chosen to compute the n optimal features for each target of the model, and then take the union over these resulting feature sets.

In practice, n was set to 150 (this choice is explained in appendix C), and the temperature interval to 50K, which is close enough to represent all temperatures but far enough to speed up the process. As the dataset contains temperatures from 5K to 800K, this corresponds to sixteen times four features sets, which totals to 64 features sets of size 150. After taking the union, this reduces to a final set of 370 features, which forms the input vector of our model.

Finally, both the input features and output targets are scaled following a min-max approach, such that they are all situated between 0 and 1. For the input features this enables to treat all features, that are sometimes very different in nature, on the same ground. Moreover, it speeds up the learning process. Concerning, the output features, this scaling is very important as the different properties can vary significantly in their order of magnitude. If no scaling was performed, the loss-function would neglect properties small in magnitude while mainly optimizing properties large in magnitude.

5.5.3 Architecture and hyperparameters

The TTNN will follow the tree-like architecture, presented in figure 5.4.5, with three levels: a first order general (thermodynamic) level splitted into the four properties: (i) vibrational entropy, (ii) Helmoltz free energy, (iii) heat capacity and (iv) the phonon contribution to the internal energy, which forms four independent second order levels connected to the first one. A third and final order level comprises the different temperatures at an interval of 20K. The complete architecture with details about the layers is depicted in figure 5.5.1, while the remaining hyperparameters are listed in table 5.5.1. Batch normalization layers are used as this speeds up the training process and moreover reduces the overall validation error.

Hyperparameter	**Value**
Optimizer	Adam (β_1=0.9, β_2=0.999, decay = 0)
Learning rate	0.006
Batch size	200
Activation function	ReLU (for each hidden layer)
Loss function	MAE
Epochs	290
Architecture	See Fig. 5.5.1.

Table 5.5.1: Hyperparameters of the TTNN.

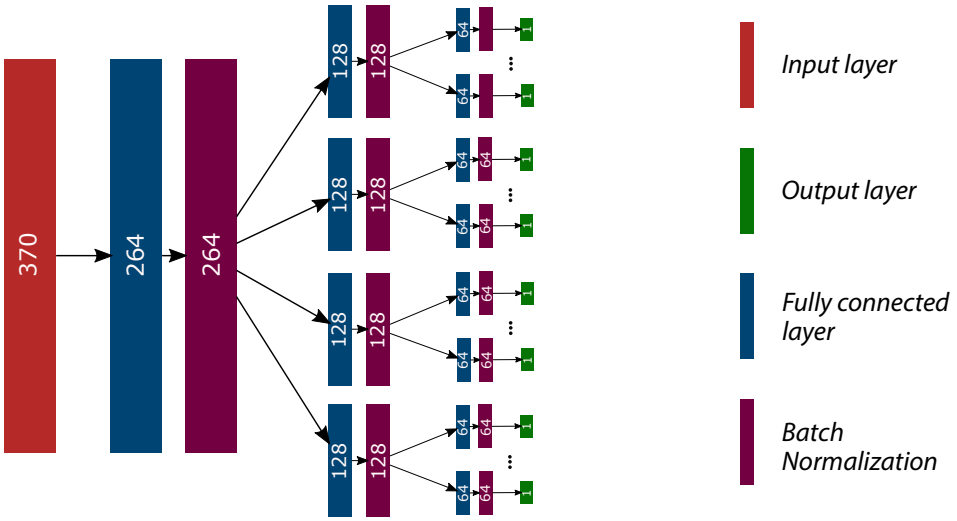

Figure 5.5.1: Schematic illustration of the architecture of the TTNN. The four divisions at the second level correspond to the four thermal properties. The divisions at the third level correspond to the different temperature. A rectified linear unit (ReLU) function is used between all layers. The annotated numbers represent the dimensionality (number of units) of the corresponding layers.

Ideally, one wants to have all targets wired into the model in order to increase the accuracy thanks to transfer learning. This would correspond to 640 targets, each at a distance of 5K. Unfortunately, this would require much more computational power than presently available in order to keep the training and optimization in a reasonable time frame. Nevertheless, the current model is already very effective as will be seen, and it can in principle only perform better by increasing the number of common targets.

The hyperparameters were chosen as the result of a simple optimization strategy consisting of a 5-fold cross validation on a grid search. Details about this are found in appendix C. As main conclusion, 3 hidden layers with 264, 128 and 16 hidden neurons as well as 150 features per target (before union) were found to be optimal.

5.5.4 Performance assessment

The TTNN performance is assessed on an independent hold-out test set, containing 155 randomly chosen samples, that has not been seen by the model during the 5-fold cross validation on the hyperparameters. This enables one to have an unbiased estimation of the model final performance. When trained on the remaining 1090 samples, it takes approximately 30 minutes to train the model on a Tesla K80 GPU.

The resulting mean absolute error for various properties are listed in table 5.5.2 . Three other algorithms were trained on the same data for comparison:

- **RF-comp.** This is the model proposed by Legrain *et al.*, see chapter 2, where a random forest containing 500 trees is used on the composition alone.
- **RF-all.** As an extension to the previous algorithm, a random forest containing 500 trees is used on all possible features from appendix D.

- **SNN.** A simple one layer neural network, as presented in section 4.6, using 200 optimal features from the RR-algorithm and a single target. This will mainly enable us to quantify the performance gained by the transfer learning principle.
- **CGCNN.** The Crystal Graph Convolutional Neural Network, from Xie *et al.* [33]. It is presented in more details in Chapter 2.

		MAE				
Property	**Typical values**	**TTNN**	**RF-comp**	**RF-all**	**SNN**	**CGCNN**
S_{vib}^{105K} [meV/K/atom]	0.0 to 0.4	0.0102	0.0293	0.0155	-	-
S_{vib}^{300K} [meV/K/atom]	0.0 to 0.7	0.0199	0.0693	0.0301	0.0248	0.0652
S_{vib}^{505K} [meV/K/atom]	0.0 to 0.8	0.0253	0.0925	0.0354	-	-
$C_{v}^{1}{}^{05K}$ [meV/K/atom]	0.0 to 2.2	0.0691	0.142	0.0806	-	-
C_{v}^{305K} [meV/K/atom]	0.1 to 3.3	0.0719	0.192	0.0779	0.0914	-
C_{v}^{505K} [meV/K/atom]	0.1 to 3.6	0.0732	0.208	0.0752	-	-
F^{105K} [meV/atom]	-16 to 158	4.38	8.91	5.96	-	-
F^{305K} [meV/atom]	-132 to 155	6.17	11.81	7.76	7.54	-
F^{505K} [meV/atom]	-285 to 142	9.41	24.58	13.68	-	-
U_{vib}^{105K} [meV/atom]	5 to 158	4.02	10.62	5.51	-	-
U_{vib}^{305K} [meV/atom]	12 to 174	5.27	18.29	6.50	6.72	- 15.98
U_{vib}^{505K} [meV/atom]	15 to 203	7.02	28.02	7.31	-	-

Table 5.5.2: Mean absolute error (MAE) for various thermodynamic properties predicted by the TTNN proposed in this work. Other methods from the literature were trained on the same data and the corresponding MAE is shown for comparison. Typical values for the different properties are listed in order to confront the magnitude of the MAE.

As can be seen from the results, the TTNN outperforms systematically all other algorithms, although all being trained on the same data. The random forest based on the composition alone performs quite poor, and this simple model proposed by Legrain *et al.* [32] shows its limitations on larger datasets. Interestingly, the orignal paper see Ref. [32], announces an MAE of 0.0367 meV/K/atom while here an MAE of 0.0693 meV/K/atom was found by reproducing their method. This bias can in principle only be due to the difference in test set. The *RF-all* model performs much better than the *RF-comp* and confirms that the composition alone is not enough. It is however not as good as the TTNN except for the specific heat where both algorithm have a similar MAE. This latter could indicate that the TTNN is not using its full potential for the specific heat and optimizing this (by adding hidden neurons for the specific heat) could be a future improvement. Nevertheless, the TTNN outperforms the *RF-all* by 40% on average, which is surprisingly good.

The *SNN* model confirms the added-value from the *tranfer principle* as the error is lowered by approximately 15% to 40% on most properties when going from the *SNN* to the TTNN.

Finally, the CGCNN is performing poorly on the two tests that were performed. Altough it was tried to optimize the hyperparameters of this model, no test MAE below 0.0652 meV/K/atom (resp. 15.98 meV/atom) was found on S_{vib}^{300K} (resp. U_{vib}^{305K}). This clearly shows the limitations of deep neural networks that do not take advantage of transfer learning when trained on a significant smaller dataset. This confirms the discussion on the bias-variance trade-off.

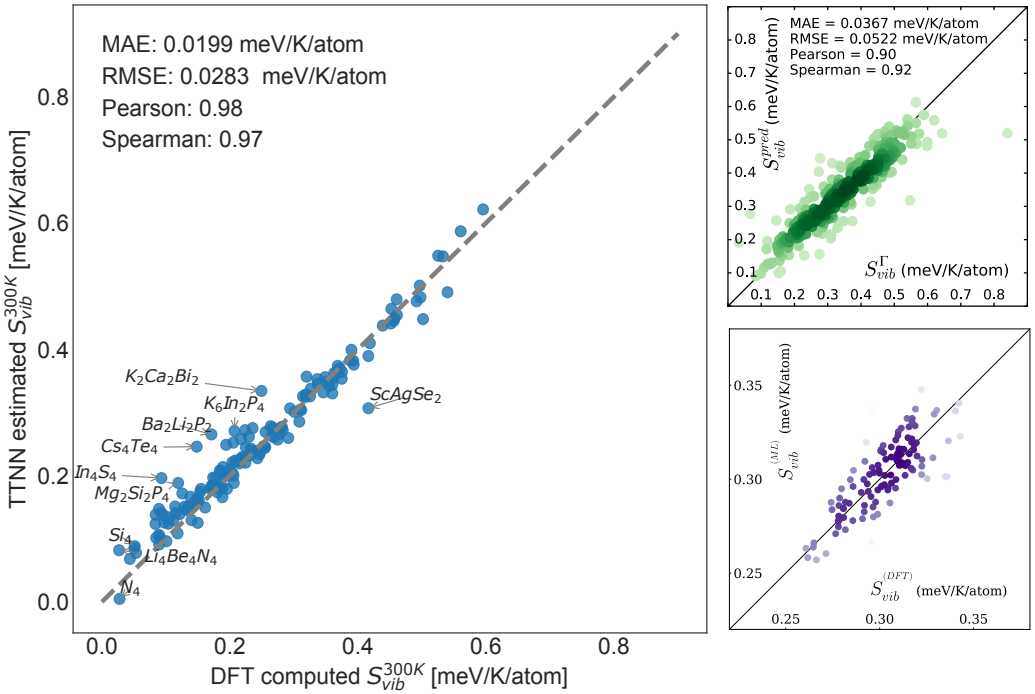

Figure 5.5.2: *Left:* TTN predicted vibrational entropy at 300K *vs.* DFT computed value. Some less well predicted entropies are annotated with their chemical formula. *Top right:* Comparison with the "composition only" model from Legrain *et al.* [32]. Image taken from Ref. [32]. *Bottom right:* Comparison with the tensorial IFC model of Legrain *et al.* [105]. Image taken from Ref. [105]. Note that the three models were validated on different datasets, but they nevertheless give a good indication of their respective performance.

Finally, concerning the vibrational entropy at 300 K, figure 5.5.2 (left) depicts the predicted value by the TTNN in function of the DFT-computed value for the previous hold-out set. A similar graph was provided for the models of Legrain *et al.* [32], [105], see chapter 2, and are depicted next too each other for a visual comparison. It is seen how the TTNN outperforms the two other models by having an approximately two times lower error.

5.6 Future improvements

The TTNN is a promising model considering the previous results. Therefore, some further improvements are discussed here. Nevertheless, the current model, as available in Ref. [106] is certainly already an excellent tool for vibrational thermal property predictions and can be used as such.

A first improvement would be to add the complete set of available thermal targets by simply reducing the temperature interval from 20 to 5 Kelvin. Another improvements would be to optimize the architecture even further. For example, it was already stated that increasing the number of hidden layers for the specific heat could be beneficial. Other activation functions could be tried too.

Finally, extending the presented idea on even more properties, such as schematically presented in figure 5.4.5 should certainly be valuable. One could imagine to have a giant

network that predicts a myriad of properties. In order to make this possible, special attention should be put on (i) the training time that could become unfeasible and (ii) how to deal with various datasets and properties. A solution to both problems would be the "learning without forgetting" paradigm, introduced by Li *et al.*, see Ref. [104]. The idea is to impute missing values by computing them with the 'old' layers beforehand and include them in the loss metric in order to learn without forgetting previously learned knowledge.

5.7 Conclusion

This chapter introduced a novel approach for predicting material properties based on a smart feature selection algorithm and transfer learning. A tree-like neural network was built that captures inter-property knowledge in a given database. This is opposed to more conventional methods where properties are seen as independent entities, with corresponding independent ML-algorithms. It was shown to outperform previous models (Refs. [32] and [105]) by providing a mean absolute error of three times as low on our benchmarks. The trained model is available in Ref. [106] and can be used as such for a fast and effective estimation of thermal data on crystalline compounds. Finally, future improvements were enumerated in order to use the presented concepts on a larger scale in the material science field.

Conclusion

We are currently witnessing a data deluge in material science, thanks to recent advances in experimental and computational methods. It is therefore crucial to process this amount of data effectively, by means of machine learning in order to advance current research. This is commonly referred to as the fourth paradigm in science. On the one hand machine learning offers new insights into materials and their properties by extracting knowledge that was not seen or captures yet. On the other hand, machine learning offers regression and classification algorithms that enables one to predict properties on-the-fly, order of magnitudes faster than conventional methods. With this motivation in mind, this dissertation applied various ML algorithms on vibrational properties of crystalline solids, key for accurate stability prediction among others.

The literature study showed us the state-of-the-art in this field, together with its current limitations. It was seen how most models are often structure and target specific, which intrinsically limits the generalization space. They were therefore labeled as 'ad-hoc'-methods. At the other side, some very general models based on convolutional neural networks have very recently been published, with promising results. However, they require often too much data than available in order to be accurate enough, at least for vibrational properties. This naturally give rise to the question, *how can one, as accurate as possible, predict properties on a large scale, given many but limited datasets containing various properties ?* This initiated the motivation of the TTNN presented in Chapter 5.

In order to build this model, and more generally any material model, one has to transform a compound structure, which is a rather complex an very raw object, into a representative fixed-length vector. The approach that was followed here can in fact be separated in two successive transformation. Firstly, instead of coming up with new descriptors, we rather simply choose a bunch of existing representative descriptors, following the RR-algorithm, presented in Chapter 3. It was based on optimizing the relevance-redundancy ratio, and this dynamically in function of the feature set size. Moreover, feature selection has the advantage of offering physical interpretation. In short the raw material was transformed into a vector capturing most possible information, constrained by relevance, redundancy, dimensionality and interpretability. In addition, the genome encoder, i.e the first common block of the TTNN presented in Chapter 5, can be seen as a second transformation where the physical features are transformed into a compact (264 size vector) representation, which forms the genome of the material.

Furthermore, next to this quantitative analysis, the NMI-grams presented in Chapter 4 brought many qualitative insights. For instance, it was seen how radial isotropic measures such as the radial distribution function are important factors for determining the vibrational entropy. This analyses also showed us the importance of convergence of NMI-grams. They enables us to identify rare features as well as to see how many samples are needed in order

to find relations/patterns between variables. For example, bond fractions, some element fractions and exotic local environment were identified as rare for datasets below 8,000 samples and should thus be avoided as features.

The final chapter proposed, based on the RR-algorithm and transfer learning, a novel model that takes advantage of information stored across different properties. Especially, this dissertation showed that transfer learning can be beneficially applied on properties that have a common nature, in order to improve the accuracy up to 30%. Compared to existing methods, the presented method provided a mean absolute error of three times as low on our benchmarks.

Finally, it should be stated that this study should not stop here. The idea behind the TTNN is powerful, and can certainly be applied on a wider scale in material science.

Appendix A

NMI convergence study

This appendix contains some supplementary figures representing the convergence of the various NMI-grams presented in Chapter 4.

A.1 Intra composition

A.2 Intra structure

A.3 Intra site

A.4 Inter structure-composition

A.5 Inter site-composition

A.6 Inter site-structure

Appendix B

Complementary figures on the compound distribution of the phonon dataset.

This appendix contains some complementary statistics, in form of histograms, about the phonon dataset (Ref. [11]) presented in section 5.2.

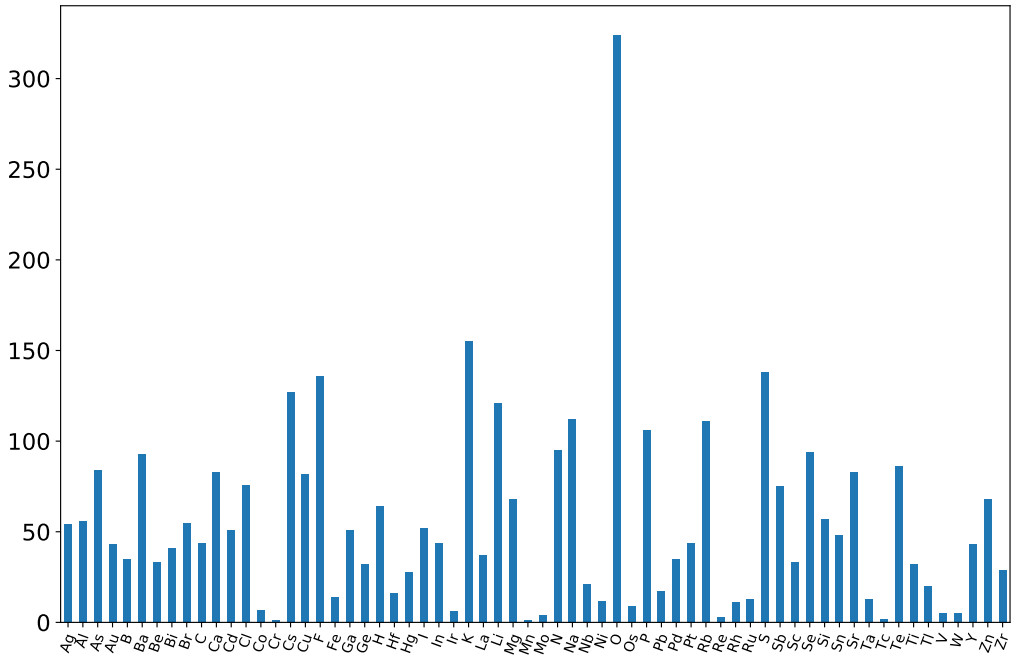

Figure B.0.1: Distribution of the elements. The count is computed as the number of compounds containing a particular element.

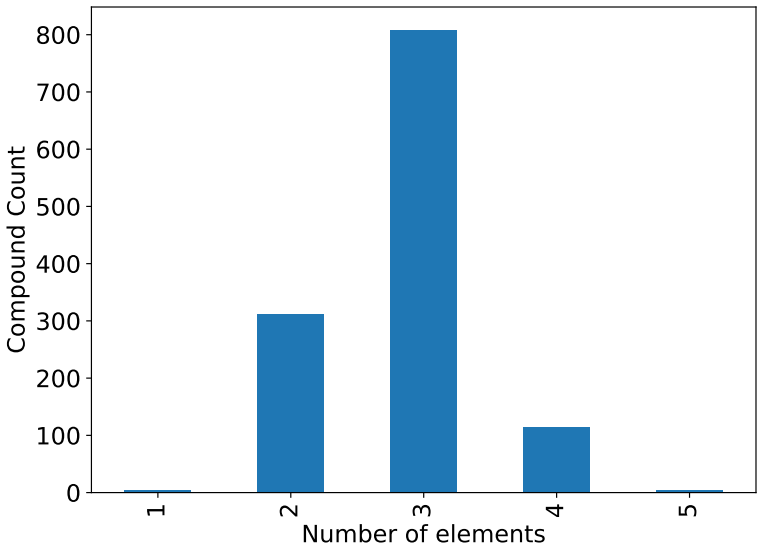

Figure B.0.2: Distribution of the number of elements per compound.

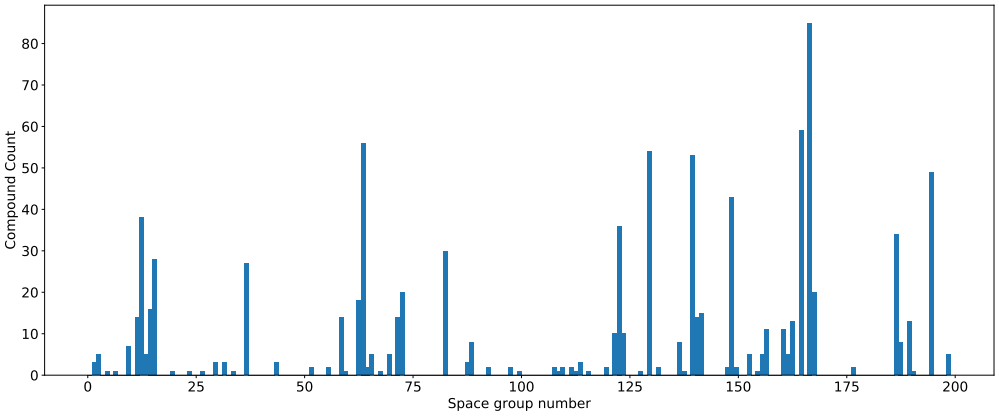

Figure B.0.3: Space group distribution.

Appendix C

TTNN hyperparameter optimization

The architecture of the TTNN is depicted in figure 5.5.1. The following notations are defined:

- L_1: The number of hidden layers in the first order block, i.e the genome encoder.
- N_1: The number of hidden units in each first order hidden layer.
- L_2: The number of hidden layers in the second order block, i.e the property-class representation.
- N_2: The number of hidden units in each first order hidden layer.
- L_3: The number of hidden layers in the third order block, i.e the intermediate temperature representation. If this is set to zero, no hidden layer is used and each property-temperature target is directly predicted from the second layer without an intermediate hidden layer.
- N_2: The number of hidden units in each third order hidden layer.

The optimization is split in three parts. First the architecture (in terms of number of hidden layers and number of neurons) is first optimized by fixing all other hyperparameters. Then, once the architecture is optimized, the number of input features per single target (i.e. before taking the union) is varied and optimized too. Finally, once the previous parameters are fixed, the batch size and learning rate are optimized.

All errors are measured on a 5-fold cross validation. The samples for this cross-validation scheme consists of all samples except 155 that were left out. These 155 left-out samples forms the test set that will be used to asses the final accuracy *after* the hyperparamter optimization, in order to avoid any bias in the estimation.

Architecture

Because making a grid-search on a set of hyperparameters is time consuming, let us first try to limit the parameters to those that will have the highest impact. From common sense, having a high number of layers L_1 and L_1 will probably not be beneficial, thus let us fix it to 1, as it should be in theory enough to model any continuous function. However

for L_3, one could argue if it is really necessary to have an extra intermediate layer, as no bifurcations are made on it. We therefore chose to give it two possible values: 0 or 1.

This fixes the number of layers. Concerning the number of neurons, we choose to let each layer take any of the following values: 32,64,128 or 264 neurons. Concerning the activation unit, the rectified linear unit function (reLU) is used and will not be changed, to avoid adding an extra dimension in the hyperparameter space. The batch size is fixed to 100, the learning rate to 0.01 and the feature set per target (before union) is set to 200.

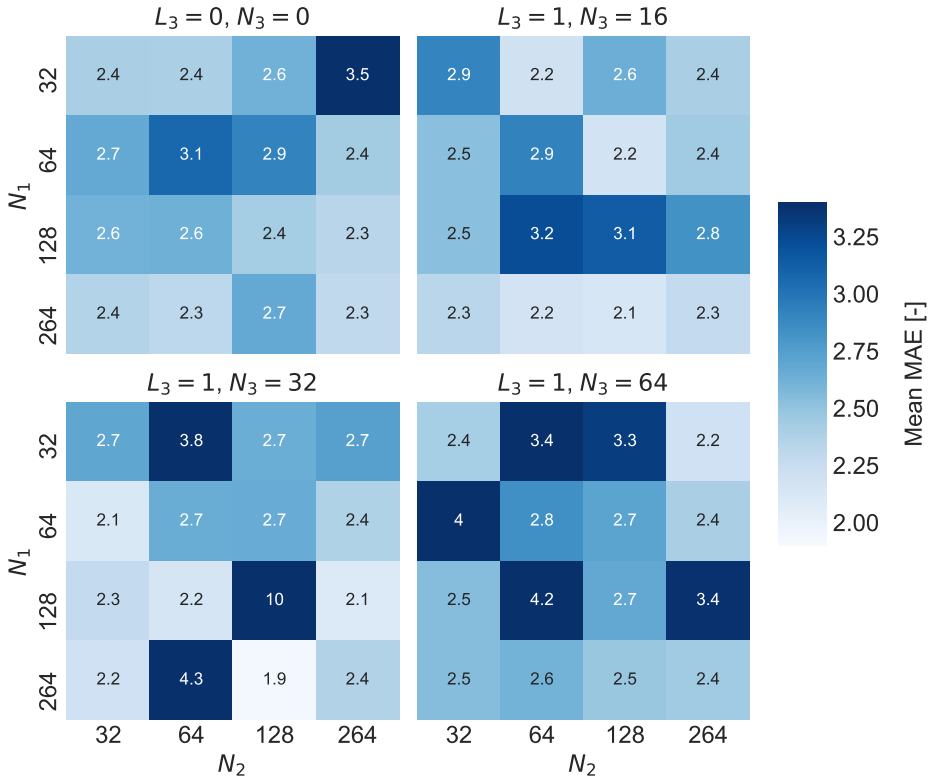

Figure C.0.1: Mean absolute error over all targets of the TTNN in function of the architecture (no units as it averages different normalized properties). A minimum is seen at $(N_1, N_2, N_3) = (264, 128, 32)$.

Once all possible combinations are defined, we simply fit our TTNN following a 5-fold cross validation scheme on each combination. The combination with the lowest 5-fold validation error will be chosen as final architecture. The results are depicted in figure C.0.1 . As can be seen, no clear tendency is found but taking (N_1, N_2, N_3) as $(264, 128, 32)$ is seen to be optimal. These values are following common sense as a high dimensionality is needed to encode to genome, while less when considering individual properties and temperatures.

Number of features

Now that final architecture has been decided, let us see if increasing or decreasing the number of optimal properties has an impact on the cross-fold validation MAE. This is done by varying the number of features chosen by the RR-algorithm on each individual target, before taking the union (see section 5.5.2 to see how to feature space is constructed). The resulting MAE is depicted in figure C.0.2.

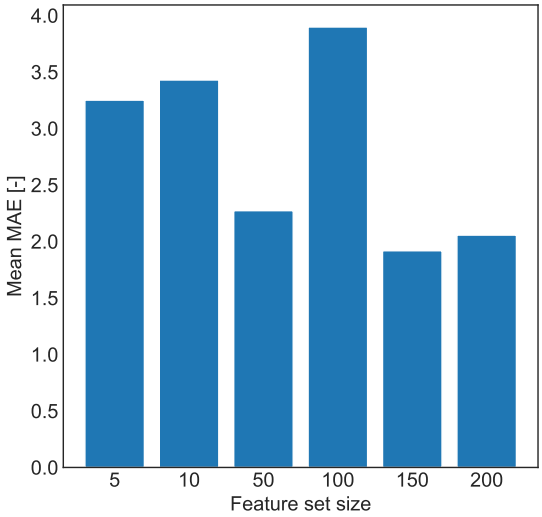

Figure C.0.2: Mean absolute error over all targets of the TTNN in function of the number of optimal features per property (before taking the union). The architecture was fixed beforehand.

From these results, it is chosen to decrease the number of features from 200 to 150. Additionally, this graph shows us that overfitting is present here when taking too many features.

Batch size and learning rate

By performing a similar grid search as before on the batch size and learning rate (taking 64, 128, 200, 500 for the batch size and 0.001, 0.005, 0.006, 0.01 for the learning rate) it was found that taking a batch size of 200 with a learning rate of 0.006 resulted in the lowest 5-fold validation.

Conclusion

The different hyperparameters were sequentially optimized in three steps. From the results, it is chosen to fix $L_1 = 1$, $N_1 = 264$, $L_2 = 1$, $N_2 = 128$, $L_3 = 1$, $N_1 = 31$, the number of features to 150 (before union), the batch size to 200 and the learning rate to 0.006.

Appendix D

List of features

The labels of the features as generated from the classes presented in chapter 3 are listed here. They are formatted as follow: IDENTIFIER: CLASS, FEATURE NAME.

D.1 Composition

C000 : AtomicOrbitals, HOMO_character
C001 : AtomicOrbitals, HOMO_element
C002 : AtomicOrbitals, HOMO_energy
C003 : AtomicOrbitals, LUMO_character
C004 : AtomicOrbitals, LUMO_element
C005 : AtomicOrbitals, LUMO_energy
C006 : AtomicOrbitals, gap_AO
C007 : AtomicPackingEfficiency, dist from 1 clusters $\|APE\| < 0.010$
C008 : AtomicPackingEfficiency, dist from 3 clusters $\|APE\| < 0.010$
C009 : AtomicPackingEfficiency, dist from 5 clusters $\|APE\| < 0.010$
C010 : AtomicPackingEfficiency, mean abs simul. packing efficiency
C011 : AtomicPackingEfficiency, mean simul. packing efficiency
C012 : BandCenter, band center
C013 : ElectronegativityDiff, maximum EN difference
C014 : ElectronegativityDiff, mean EN difference
C015 : ElectronegativityDiff, minimum EN difference
C016 : ElectronegativityDiff, range EN difference
C017 : ElectronegativityDiff, std_dev EN difference
C018 : ElementProperty, MagpieData avg_dev AtomicWeight
C019 : ElementProperty, MagpieData avg_dev Column
C020 : ElementProperty, MagpieData avg_dev CovalentRadius
C021 : ElementProperty, MagpieData avg_dev Electronegativity
C022 : ElementProperty, MagpieData avg_dev GSbandgap
C023 : ElementProperty, MagpieData avg_dev GSmagmom
C024 : ElementProperty, MagpieData avg_dev GSvolume_pa
C025 : ElementProperty, MagpieData avg_dev MeltingT
C026 : ElementProperty, MagpieData avg_dev MendeleevNumber
C027 : ElementProperty, MagpieData avg_dev NUnfilled
C028 : ElementProperty, MagpieData avg_dev NValence

C029 : ElementProperty, MagpieData avg_dev NdUnfilled
C030 : ElementProperty, MagpieData avg_dev NdValence
C031 : ElementProperty, MagpieData avg_dev NfUnfilled
C032 : ElementProperty, MagpieData avg_dev NfValence
C033 : ElementProperty, MagpieData avg_dev NpUnfilled
C034 : ElementProperty, MagpieData avg_dev NpValence
C035 : ElementProperty, MagpieData avg_dev NsUnfilled
C036 : ElementProperty, MagpieData avg_dev NsValence
C037 : ElementProperty, MagpieData avg_dev Number
C038 : ElementProperty, MagpieData avg_dev Row
C039 : ElementProperty, MagpieData avg_dev SpaceGroupNumber
C040 : ElementProperty, MagpieData maximum AtomicWeight
C041 : ElementProperty, MagpieData maximum Column
C042 : ElementProperty, MagpieData maximum CovalentRadius
C043 : ElementProperty, MagpieData maximum Electronegativity
C044 : ElementProperty, MagpieData maximum GSbandgap
C045 : ElementProperty, MagpieData maximum GSmagmom
C046 : ElementProperty, MagpieData maximum GSvolume_pa
C047 : ElementProperty, MagpieData maximum MeltingT
C048 : ElementProperty, MagpieData maximum MendeleevNumber
C049 : ElementProperty, MagpieData maximum NUnfilled
C050 : ElementProperty, MagpieData maximum NValence
C051 : ElementProperty, MagpieData maximum NdUnfilled
C052 : ElementProperty, MagpieData maximum NdValence
C053 : ElementProperty, MagpieData maximum NfUnfilled
C054 : ElementProperty, MagpieData maximum NfValence
C055 : ElementProperty, MagpieData maximum NpUnfilled
C056 : ElementProperty, MagpieData maximum NpValence
C057 : ElementProperty, MagpieData maximum NsUnfilled
C058 : ElementProperty, MagpieData maximum NsValence
C059 : ElementProperty, MagpieData maximum Number
C060 : ElementProperty, MagpieData maximum Row
C061 : ElementProperty, MagpieData maximum SpaceGroupNumber
C062 : ElementProperty, MagpieData mean AtomicWeight
C063 : ElementProperty, MagpieData mean Column
C064 : ElementProperty, MagpieData mean CovalentRadius
C065 : ElementProperty, MagpieData mean Electronegativity
C066 : ElementProperty, MagpieData mean GSbandgap
C067 : ElementProperty, MagpieData mean GSmagmom
C068 : ElementProperty, MagpieData mean GSvolume_pa
C069 : ElementProperty, MagpieData mean MeltingT
C070 : ElementProperty, MagpieData mean MendeleevNumber
C071 : ElementProperty, MagpieData mean NUnfilled
C072 : ElementProperty, MagpieData mean NValence
C073 : ElementProperty, MagpieData mean NdUnfilled
C074 : ElementProperty, MagpieData mean NdValence
C075 : ElementProperty, MagpieData mean NfUnfilled
C076 : ElementProperty, MagpieData mean NfValence

C077 : ElementProperty, MagpieData mean NpUnfilled
C078 : ElementProperty, MagpieData mean NpValence
C079 : ElementProperty, MagpieData mean NsUnfilled
C080 : ElementProperty, MagpieData mean NsValence
C081 : ElementProperty, MagpieData mean Number
C082 : ElementProperty, MagpieData mean Row
C083 : ElementProperty, MagpieData mean SpaceGroupNumber
C084 : ElementProperty, MagpieData minimum AtomicWeight
C085 : ElementProperty, MagpieData minimum Column
C086 : ElementProperty, MagpieData minimum CovalentRadius
C087 : ElementProperty, MagpieData minimum Electronegativity
C088 : ElementProperty, MagpieData minimum GSbandgap
C089 : ElementProperty, MagpieData minimum GSmagmom
C090 : ElementProperty, MagpieData minimum GSvolume_pa
C091 : ElementProperty, MagpieData minimum MeltingT
C092 : ElementProperty, MagpieData minimum MendeleevNumber
C093 : ElementProperty, MagpieData minimum NUnfilled
C094 : ElementProperty, MagpieData minimum NValence
C095 : ElementProperty, MagpieData minimum NdUnfilled
C096 : ElementProperty, MagpieData minimum NdValence
C097 : ElementProperty, MagpieData minimum NfUnfilled
C098 : ElementProperty, MagpieData minimum NfValence
C099 : ElementProperty, MagpieData minimum NpUnfilled
C100 : ElementProperty, MagpieData minimum NpValence
C101 : ElementProperty, MagpieData minimum NsUnfilled
C102 : ElementProperty, MagpieData minimum NsValence
C103 : ElementProperty, MagpieData minimum Number
C104 : ElementProperty, MagpieData minimum Row
C105 : ElementProperty, MagpieData minimum SpaceGroupNumber
C106 : ElementProperty, MagpieData mode AtomicWeight
C107 : ElementProperty, MagpieData mode Column
C108 : ElementProperty, MagpieData mode CovalentRadius
C109 : ElementProperty, MagpieData mode Electronegativity
C110 : ElementProperty, MagpieData mode GSbandgap
C111 : ElementProperty, MagpieData mode GSmagmom
C112 : ElementProperty, MagpieData mode GSvolume_pa
C113 : ElementProperty, MagpieData mode MeltingT
C114 : ElementProperty, MagpieData mode MendeleevNumber
C115 : ElementProperty, MagpieData mode NUnfilled
C116 : ElementProperty, MagpieData mode NValence
C117 : ElementProperty, MagpieData mode NdUnfilled
C118 : ElementProperty, MagpieData mode NdValence
C119 : ElementProperty, MagpieData mode NfUnfilled
C120 : ElementProperty, MagpieData mode NfValence
C121 : ElementProperty, MagpieData mode NpUnfilled
C122 : ElementProperty, MagpieData mode NpValence
C123 : ElementProperty, MagpieData mode NsUnfilled
C124 : ElementProperty, MagpieData mode NsValence

C125 : ElementProperty, MagpieData mode Number
C126 : ElementProperty, MagpieData mode Row
C127 : ElementProperty, MagpieData mode SpaceGroupNumber
C128 : ElementProperty, MagpieData range AtomicWeight
C129 : ElementProperty, MagpieData range Column
C130 : ElementProperty, MagpieData range CovalentRadius
C131 : ElementProperty, MagpieData range Electronegativity
C132 : ElementProperty, MagpieData range GSbandgap
C133 : ElementProperty, MagpieData range GSmagmom
C134 : ElementProperty, MagpieData range GSvolume_pa
C135 : ElementProperty, MagpieData range MeltingT
C136 : ElementProperty, MagpieData range MendeleevNumber
C137 : ElementProperty, MagpieData range NUnfilled
C138 : ElementProperty, MagpieData range NValence
C139 : ElementProperty, MagpieData range NdUnfilled
C140 : ElementProperty, MagpieData range NdValence
C141 : ElementProperty, MagpieData range NfUnfilled
C142 : ElementProperty, MagpieData range NfValence
C143 : ElementProperty, MagpieData range NpUnfilled
C144 : ElementProperty, MagpieData range NpValence
C145 : ElementProperty, MagpieData range NsUnfilled
C146 : ElementProperty, MagpieData range NsValence
C147 : ElementProperty, MagpieData range Number
C148 : ElementProperty, MagpieData range Row
C149 : ElementProperty, MagpieData range SpaceGroupNumber
C150 : Input Data, bulk modulus
C151 : Input Data, e_form
C152 : Input Data, e_hull
C153 : Input Data, elastic anisotropy
C154 : Input Data, gap pbe
C155 : Input Data, mu_b
C156 : Input Data, shear modulus
C157 : IonProperty, avg ionic char
C158 : IonProperty, max ionic char
C159 : Miedema, Miedema_deltaH_inter
C160 : OxidationStates, maximum oxidation state
C161 : OxidationStates, minimum oxidation state
C162 : OxidationStates, range oxidation state
C163 : OxidationStates, std_dev oxidation state
C164 : Stoichiometry, 0-norm
C165 : Stoichiometry, 10-norm
C166 : Stoichiometry, 2-norm
C167 : Stoichiometry, 3-norm
C168 : Stoichiometry, 5-norm
C169 : Stoichiometry, 7-norm
C170 : TMetalFraction, transition metal fraction
C171 : ValenceOrbital, avg d valence electrons
C172 : ValenceOrbital, avg f valence electrons

C173 : ValenceOrbital, avg p valence electrons
C174 : ValenceOrbital, avg s valence electrons
C175 : ValenceOrbital, frac d valence electrons
C176 : ValenceOrbital, frac f valence electrons
C177 : ValenceOrbital, frac p valence electrons
C178 : ValenceOrbital, frac s valence electrons
C179 : YangSolidSolution, Yang delta
C180 : YangSolidSolution, Yang omega
C181 : IonProperty, compound possible
C182 : ElementFraction, Al
C183 : ElementFraction, B
C184 : ElementFraction, Be
C185 : ElementFraction, C
C186 : ElementFraction, Ca
C187 : ElementFraction, Cl
C188 : ElementFraction, Co
C189 : ElementFraction, Cr
C190 : ElementFraction, Cu
C191 : ElementFraction, F
C192 : ElementFraction, Fe
C193 : ElementFraction, He
C194 : ElementFraction, K
C195 : ElementFraction, Li
C196 : ElementFraction, Mg
C197 : ElementFraction, Mn
C198 : ElementFraction, N
C199 : ElementFraction, Na
C200 : ElementFraction, Ne
C201 : ElementFraction, Ni
C202 : ElementFraction, O
C203 : ElementFraction, P
C204 : ElementFraction, S
C205 : ElementFraction, Sc
C206 : ElementFraction, Si
C207 : ElementFraction, Ti
C208 : ElementFraction, V
C209 : ElementFraction, Zn

D.2 Structure

S000 : ChemicalOrdering, mean ordering parameter shell 1
S001 : ChemicalOrdering, mean ordering parameter shell 2
S002 : ChemicalOrdering, mean ordering parameter shell 3
S003 : CoulombMatrix, coulomb matrix eig 0
S004 : CoulombMatrix, coulomb matrix eig 1
S005 : CoulombMatrix, coulomb matrix eig 10
S006 : CoulombMatrix, coulomb matrix eig 2

S007 : CoulombMatrix, coulomb matrix eig 3
S008 : CoulombMatrix, coulomb matrix eig 4
S009 : CoulombMatrix, coulomb matrix eig 5
S010 : CoulombMatrix, coulomb matrix eig 6
S011 : CoulombMatrix, coulomb matrix eig 7
S012 : CoulombMatrix, coulomb matrix eig 8
S013 : CoulombMatrix, coulomb matrix eig 9
S014 : DensityFeatures, density
S015 : DensityFeatures, packing fraction
S016 : DensityFeatures, vpa
S017 : GlobalSymmetryFeatures, crystal_system
S018 : GlobalSymmetryFeatures, crystal_system_int
S019 : GlobalSymmetryFeatures, is_centrosymmetric
S020 : GlobalSymmetryFeatures, spacegroup_num
S021 : Input Data, bulk modulus
S022 : Input Data, e_form
S023 : Input Data, e_hull
S024 : Input Data, elastic anisotropy
S025 : Input Data, gap pbe
S026 : Input Data, mu_b
S027 : Input Data, shear modulus
S028 : MaximumPackingEfficiency, max packing efficiency
S029 : RadialDistributionFunction, radial distribution function, d_0.00
S030 : RadialDistributionFunction, radial distribution function, d_0.10
S031 : RadialDistributionFunction, radial distribution function, d_0.20
S032 : RadialDistributionFunction, radial distribution function, d_0.30
S033 : RadialDistributionFunction, radial distribution function, d_0.40
S034 : RadialDistributionFunction, radial distribution function, d_0.50
S035 : RadialDistributionFunction, radial distribution function, d_0.60
S036 : RadialDistributionFunction, radial distribution function, d_0.70
S037 : RadialDistributionFunction, radial distribution function, d_0.80
S038 : RadialDistributionFunction, radial distribution function, d_0.90
S039 : RadialDistributionFunction, radial distribution function, d_1.00
S040 : RadialDistributionFunction, radial distribution function, d_1.10
S041 : RadialDistributionFunction, radial distribution function, d_1.20
S042 : RadialDistributionFunction, radial distribution function, d_1.30
S043 : RadialDistributionFunction, radial distribution function, d_1.40
S044 : RadialDistributionFunction, radial distribution function, d_1.50
S045 : RadialDistributionFunction, radial distribution function, d_1.60
S046 : RadialDistributionFunction, radial distribution function, d_1.70
S047 : RadialDistributionFunction, radial distribution function, d_1.80
S048 : RadialDistributionFunction, radial distribution function, d_1.90
S049 : RadialDistributionFunction, radial distribution function, d_2.00
S050 : RadialDistributionFunction, radial distribution function, d_2.10
S051 : RadialDistributionFunction, radial distribution function, d_2.20
S052 : RadialDistributionFunction, radial distribution function, d_2.30
S053 : RadialDistributionFunction, radial distribution function, d_2.40
S054 : RadialDistributionFunction, radial distribution function, d_2.50

S055 : RadialDistributionFunction, radial distribution function, d_2.60
S056 : RadialDistributionFunction, radial distribution function, d_2.70
S057 : RadialDistributionFunction, radial distribution function, d_2.80
S058 : RadialDistributionFunction, radial distribution function, d_2.90
S059 : RadialDistributionFunction, radial distribution function, d_3.00
S060 : RadialDistributionFunction, radial distribution function, d_3.10
S061 : RadialDistributionFunction, radial distribution function, d_3.20
S062 : RadialDistributionFunction, radial distribution function, d_3.30
S063 : RadialDistributionFunction, radial distribution function, d_3.40
S064 : RadialDistributionFunction, radial distribution function, d_3.50
S065 : RadialDistributionFunction, radial distribution function, d_3.60
S066 : RadialDistributionFunction, radial distribution function, d_3.70
S067 : RadialDistributionFunction, radial distribution function, d_3.80
S068 : RadialDistributionFunction, radial distribution function, d_3.90
S069 : RadialDistributionFunction, radial distribution function, d_4.00
S070 : RadialDistributionFunction, radial distribution function, d_4.10
S071 : RadialDistributionFunction, radial distribution function, d_4.20
S072 : RadialDistributionFunction, radial distribution function, d_4.30
S073 : RadialDistributionFunction, radial distribution function, d_4.40
S074 : RadialDistributionFunction, radial distribution function, d_4.50
S075 : RadialDistributionFunction, radial distribution function, d_4.60
S076 : RadialDistributionFunction, radial distribution function, d_4.70
S077 : RadialDistributionFunction, radial distribution function, d_4.80
S078 : RadialDistributionFunction, radial distribution function, d_4.90
S079 : SineCoulombMatrix, sine coulomb matrix eig 0
S080 : SineCoulombMatrix, sine coulomb matrix eig 1
S081 : SineCoulombMatrix, sine coulomb matrix eig 10
S082 : SineCoulombMatrix, sine coulomb matrix eig 2
S083 : SineCoulombMatrix, sine coulomb matrix eig 3
S084 : SineCoulombMatrix, sine coulomb matrix eig 4
S085 : SineCoulombMatrix, sine coulomb matrix eig 5
S086 : SineCoulombMatrix, sine coulomb matrix eig 6
S087 : SineCoulombMatrix, sine coulomb matrix eig 7
S088 : SineCoulombMatrix, sine coulomb matrix eig 8
S089 : SineCoulombMatrix, sine coulomb matrix eig 9
S090 : StructuralHeterogeneity, avg_dev neighbor distance variation
S091 : StructuralHeterogeneity, max relative bond length
S092 : StructuralHeterogeneity, maximum neighbor distance variation
S093 : StructuralHeterogeneity, mean absolute deviation in relative bond length
S094 : StructuralHeterogeneity, mean absolute deviation in relative cell size
S095 : StructuralHeterogeneity, mean neighbor distance variation
S096 : StructuralHeterogeneity, min relative bond length
S097 : StructuralHeterogeneity, minimum neighbor distance variation
S098 : StructuralHeterogeneity, range neighbor distance variation
S099 : XRDPowderPattern, xrd_000
S100 : XRDPowderPattern, xrd_001
S101 : XRDPowderPattern, xrd_003
S102 : XRDPowderPattern, xrd_005

S103 : XRDPowderPattern, xrd_007
S104 : XRDPowderPattern, xrd_009
S105 : XRDPowderPattern, xrd_011
S106 : XRDPowderPattern, xrd_013
S107 : XRDPowderPattern, xrd_015
S108 : XRDPowderPattern, xrd_017
S109 : XRDPowderPattern, xrd_019
S110 : XRDPowderPattern, xrd_021
S111 : XRDPowderPattern, xrd_023
S112 : XRDPowderPattern, xrd_025
S113 : XRDPowderPattern, xrd_027
S114 : XRDPowderPattern, xrd_029
S115 : XRDPowderPattern, xrd_031
S116 : XRDPowderPattern, xrd_033
S117 : XRDPowderPattern, xrd_035
S118 : XRDPowderPattern, xrd_037
S119 : XRDPowderPattern, xrd_039
S120 : XRDPowderPattern, xrd_041
S121 : XRDPowderPattern, xrd_043
S122 : XRDPowderPattern, xrd_045
S123 : XRDPowderPattern, xrd_047
S124 : XRDPowderPattern, xrd_049
S125 : XRDPowderPattern, xrd_051
S126 : XRDPowderPattern, xrd_053
S127 : XRDPowderPattern, xrd_055
S128 : XRDPowderPattern, xrd_057
S129 : XRDPowderPattern, xrd_059
S130 : XRDPowderPattern, xrd_061
S131 : XRDPowderPattern, xrd_063
S132 : XRDPowderPattern, xrd_065
S133 : XRDPowderPattern, xrd_067
S134 : XRDPowderPattern, xrd_069
S135 : XRDPowderPattern, xrd_071
S136 : XRDPowderPattern, xrd_073
S137 : XRDPowderPattern, xrd_075
S138 : XRDPowderPattern, xrd_077
S139 : XRDPowderPattern, xrd_079
S140 : XRDPowderPattern, xrd_081
S141 : XRDPowderPattern, xrd_083
S142 : XRDPowderPattern, xrd_085
S143 : XRDPowderPattern, xrd_087
S144 : XRDPowderPattern, xrd_089
S145 : XRDPowderPattern, xrd_091
S146 : XRDPowderPattern, xrd_093
S147 : XRDPowderPattern, xrd_095
S148 : XRDPowderPattern, xrd_097
S149 : XRDPowderPattern, xrd_099
S150 : XRDPowderPattern, xrd_101

S151 : XRDPowderPattern, xrd_103
S152 : XRDPowderPattern, xrd_105
S153 : XRDPowderPattern, xrd_107
S154 : XRDPowderPattern, xrd_109
S155 : XRDPowderPattern, xrd_111
S156 : XRDPowderPattern, xrd_113
S157 : XRDPowderPattern, xrd_115
S158 : XRDPowderPattern, xrd_117
S159 : XRDPowderPattern, xrd_119
S160 : XRDPowderPattern, xrd_121
S161 : XRDPowderPattern, xrd_123
S162 : XRDPowderPattern, xrd_125
S163 : XRDPowderPattern, xrd_127
S164 : BondFractions, Al - Al bond frac.
S165 : BondFractions, Al - B bond frac.
S166 : BondFractions, Al - Be bond frac.
S167 : BondFractions, Al - C bond frac.
S168 : BondFractions, Al - Ca bond frac.
S169 : BondFractions, Al - Cl bond frac.
S170 : BondFractions, Al - Co bond frac.
S171 : BondFractions, Al - Cr bond frac.
S172 : BondFractions, Al - Cu bond frac.
S173 : BondFractions, Al - F bond frac.
S174 : BondFractions, Al - Fe bond frac.
S175 : BondFractions, Al - K bond frac.
S176 : BondFractions, Al - Li bond frac.
S177 : BondFractions, Al - Mg bond frac.
S178 : BondFractions, Al - Mn bond frac.
S179 : BondFractions, Al - N bond frac.
S180 : BondFractions, Al - Na bond frac.
S181 : BondFractions, Al - Ni bond frac.
S182 : BondFractions, Al - O bond frac.
S183 : BondFractions, Al - P bond frac.
S184 : BondFractions, Al - S bond frac.
S185 : BondFractions, Al - Sc bond frac.
S186 : BondFractions, Al - Si bond frac.
S187 : BondFractions, Al - Ti bond frac.
S188 : BondFractions, Al - V bond frac.
S189 : BondFractions, Al - Zn bond frac.
S190 : BondFractions, B - B bond frac.
S191 : BondFractions, B - Be bond frac.
S192 : BondFractions, B - C bond frac.
S193 : BondFractions, B - Ca bond frac.
S194 : BondFractions, B - Cl bond frac.
S195 : BondFractions, B - Co bond frac.
S196 : BondFractions, B - Cr bond frac.
S197 : BondFractions, B - Cu bond frac.
S198 : BondFractions, B - F bond frac.

S199 : BondFractions, B - Fe bond frac.
S200 : BondFractions, B - K bond frac.
S201 : BondFractions, B - Li bond frac.
S202 : BondFractions, B - Mg bond frac.
S203 : BondFractions, B - Mn bond frac.
S204 : BondFractions, B - N bond frac.
S205 : BondFractions, B - Na bond frac.
S206 : BondFractions, B - Ni bond frac.
S207 : BondFractions, B - O bond frac.
S208 : BondFractions, B - P bond frac.
S209 : BondFractions, B - S bond frac.
S210 : BondFractions, B - Sc bond frac.
S211 : BondFractions, B - Si bond frac.
S212 : BondFractions, B - Ti bond frac.
S213 : BondFractions, B - V bond frac.
S214 : BondFractions, B - Zn bond frac.
S215 : BondFractions, Be - Be bond frac.
S216 : BondFractions, Be - C bond frac.
S217 : BondFractions, Be - Ca bond frac.
S218 : BondFractions, Be - Cl bond frac.
S219 : BondFractions, Be - Co bond frac.
S220 : BondFractions, Be - Cr bond frac.
S221 : BondFractions, Be - Cu bond frac.
S222 : BondFractions, Be - F bond frac.
S223 : BondFractions, Be - Fe bond frac.
S224 : BondFractions, Be - K bond frac.
S225 : BondFractions, Be - Li bond frac.
S226 : BondFractions, Be - Mg bond frac.
S227 : BondFractions, Be - Mn bond frac.
S228 : BondFractions, Be - N bond frac.
S229 : BondFractions, Be - Na bond frac.
S230 : BondFractions, Be - Ni bond frac.
S231 : BondFractions, Be - O bond frac.
S232 : BondFractions, Be - P bond frac.
S233 : BondFractions, Be - S bond frac.
S234 : BondFractions, Be - Sc bond frac.
S235 : BondFractions, Be - Si bond frac.
S236 : BondFractions, Be - Ti bond frac.
S237 : BondFractions, Be - V bond frac.
S238 : BondFractions, Be - Zn bond frac.
S239 : BondFractions, C - C bond frac.
S240 : BondFractions, C - Ca bond frac.
S241 : BondFractions, C - Cl bond frac.
S242 : BondFractions, C - Co bond frac.
S243 : BondFractions, C - Cr bond frac.
S244 : BondFractions, C - Cu bond frac.
S245 : BondFractions, C - F bond frac.
S246 : BondFractions, C - Fe bond frac.

S247 : BondFractions, C - K bond frac.
S248 : BondFractions, C - Li bond frac.
S249 : BondFractions, C - Mg bond frac.
S250 : BondFractions, C - Mn bond frac.
S251 : BondFractions, C - N bond frac.
S252 : BondFractions, C - Na bond frac.
S253 : BondFractions, C - Ni bond frac.
S254 : BondFractions, C - O bond frac.
S255 : BondFractions, C - P bond frac.
S256 : BondFractions, C - S bond frac.
S257 : BondFractions, C - Sc bond frac.
S258 : BondFractions, C - Si bond frac.
S259 : BondFractions, C - Ti bond frac.
S260 : BondFractions, C - V bond frac.
S261 : BondFractions, C - Zn bond frac.
S262 : BondFractions, Ca - Ca bond frac.
S263 : BondFractions, Ca - Cl bond frac.
S264 : BondFractions, Ca - Co bond frac.
S265 : BondFractions, Ca - Cr bond frac.
S266 : BondFractions, Ca - Cu bond frac.
S267 : BondFractions, Ca - F bond frac.
S268 : BondFractions, Ca - Fe bond frac.
S269 : BondFractions, Ca - K bond frac.
S270 : BondFractions, Ca - Li bond frac.
S271 : BondFractions, Ca - Mg bond frac.
S272 : BondFractions, Ca - Mn bond frac.
S273 : BondFractions, Ca - N bond frac.
S274 : BondFractions, Ca - Na bond frac.
S275 : BondFractions, Ca - Ni bond frac.
S276 : BondFractions, Ca - O bond frac.
S277 : BondFractions, Ca - P bond frac.
S278 : BondFractions, Ca - S bond frac.
S279 : BondFractions, Ca - Sc bond frac.
S280 : BondFractions, Ca - Si bond frac.
S281 : BondFractions, Ca - Ti bond frac.
S282 : BondFractions, Ca - V bond frac.
S283 : BondFractions, Ca - Zn bond frac.
S284 : BondFractions, Cl - Cl bond frac.
S285 : BondFractions, Cl - Co bond frac.
S286 : BondFractions, Cl - Cr bond frac.
S287 : BondFractions, Cl - Cu bond frac.
S288 : BondFractions, Cl - F bond frac.
S289 : BondFractions, Cl - Fe bond frac.
S290 : BondFractions, Cl - K bond frac.
S291 : BondFractions, Cl - Li bond frac.
S292 : BondFractions, Cl - Mg bond frac.
S293 : BondFractions, Cl - Mn bond frac.
S294 : BondFractions, Cl - N bond frac.

S295 : BondFractions, Cl - Na bond frac.
S296 : BondFractions, Cl - Ni bond frac.
S297 : BondFractions, Cl - O bond frac.
S298 : BondFractions, Cl - P bond frac.
S299 : BondFractions, Cl - S bond frac.
S300 : BondFractions, Cl - Sc bond frac.
S301 : BondFractions, Cl - Si bond frac.
S302 : BondFractions, Cl - Ti bond frac.
S303 : BondFractions, Cl - V bond frac.
S304 : BondFractions, Cl - Zn bond frac.
S305 : BondFractions, Co - Co bond frac.
S306 : BondFractions, Co - Cr bond frac.
S307 : BondFractions, Co - Cu bond frac.
S308 : BondFractions, Co - F bond frac.
S309 : BondFractions, Co - Fe bond frac.
S310 : BondFractions, Co - K bond frac.
S311 : BondFractions, Co - Li bond frac.
S312 : BondFractions, Co - Mg bond frac.
S313 : BondFractions, Co - Mn bond frac.
S314 : BondFractions, Co - N bond frac.
S315 : BondFractions, Co - Na bond frac.
S316 : BondFractions, Co - Ni bond frac.
S317 : BondFractions, Co - O bond frac.
S318 : BondFractions, Co - P bond frac.
S319 : BondFractions, Co - S bond frac.
S320 : BondFractions, Co - Sc bond frac.
S321 : BondFractions, Co - Si bond frac.
S322 : BondFractions, Co - Ti bond frac.
S323 : BondFractions, Co - V bond frac.
S324 : BondFractions, Co - Zn bond frac.
S325 : BondFractions, Cr - Cr bond frac.
S326 : BondFractions, Cr - Cu bond frac.
S327 : BondFractions, Cr - F bond frac.
S328 : BondFractions, Cr - Fe bond frac.
S329 : BondFractions, Cr - K bond frac.
S330 : BondFractions, Cr - Li bond frac.
S331 : BondFractions, Cr - Mg bond frac.
S332 : BondFractions, Cr - Mn bond frac.
S333 : BondFractions, Cr - N bond frac.
S334 : BondFractions, Cr - Na bond frac.
S335 : BondFractions, Cr - Ni bond frac.
S336 : BondFractions, Cr - O bond frac.
S337 : BondFractions, Cr - P bond frac.
S338 : BondFractions, Cr - S bond frac.
S339 : BondFractions, Cr - Sc bond frac.
S340 : BondFractions, Cr - Si bond frac.
S341 : BondFractions, Cr - Ti bond frac.
S342 : BondFractions, Cr - V bond frac.

S343 : BondFractions, Cr - Zn bond frac.
S344 : BondFractions, Cu - Cu bond frac.
S345 : BondFractions, Cu - F bond frac.
S346 : BondFractions, Cu - Fe bond frac.
S347 : BondFractions, Cu - K bond frac.
S348 : BondFractions, Cu - Li bond frac.
S349 : BondFractions, Cu - Mg bond frac.
S350 : BondFractions, Cu - Mn bond frac.
S351 : BondFractions, Cu - N bond frac.
S352 : BondFractions, Cu - Na bond frac.
S353 : BondFractions, Cu - Ni bond frac.
S354 : BondFractions, Cu - O bond frac.
S355 : BondFractions, Cu - P bond frac.
S356 : BondFractions, Cu - S bond frac.
S357 : BondFractions, Cu - Sc bond frac.
S358 : BondFractions, Cu - Si bond frac.
S359 : BondFractions, Cu - Ti bond frac.
S360 : BondFractions, Cu - V bond frac.
S361 : BondFractions, Cu - Zn bond frac.
S362 : BondFractions, F - F bond frac.
S363 : BondFractions, F - Fe bond frac.
S364 : BondFractions, F - K bond frac.
S365 : BondFractions, F - Li bond frac.
S366 : BondFractions, F - Mg bond frac.
S367 : BondFractions, F - Mn bond frac.
S368 : BondFractions, F - N bond frac.
S369 : BondFractions, F - Na bond frac.
S370 : BondFractions, F - Ni bond frac.
S371 : BondFractions, F - O bond frac.
S372 : BondFractions, F - P bond frac.
S373 : BondFractions, F - S bond frac.
S374 : BondFractions, F - Sc bond frac.
S375 : BondFractions, F - Si bond frac.
S376 : BondFractions, F - Ti bond frac.
S377 : BondFractions, F - V bond frac.
S378 : BondFractions, F - Zn bond frac.
S379 : BondFractions, Fe - Fe bond frac.
S380 : BondFractions, Fe - K bond frac.
S381 : BondFractions, Fe - Li bond frac.
S382 : BondFractions, Fe - Mg bond frac.
S383 : BondFractions, Fe - Mn bond frac.
S384 : BondFractions, Fe - N bond frac.
S385 : BondFractions, Fe - Na bond frac.
S386 : BondFractions, Fe - Ni bond frac.
S387 : BondFractions, Fe - O bond frac.
S388 : BondFractions, Fe - P bond frac.
S389 : BondFractions, Fe - S bond frac.
S390 : BondFractions, Fe - Sc bond frac.

S391 : BondFractions, Fe - Si bond frac.
S392 : BondFractions, Fe - Ti bond frac.
S393 : BondFractions, Fe - V bond frac.
S394 : BondFractions, Fe - Zn bond frac.
S395 : BondFractions, He - He bond frac.
S396 : BondFractions, K - K bond frac.
S397 : BondFractions, K - Li bond frac.
S398 : BondFractions, K - Mg bond frac.
S399 : BondFractions, K - Mn bond frac.
S400 : BondFractions, K - N bond frac.
S401 : BondFractions, K - Na bond frac.
S402 : BondFractions, K - Ni bond frac.
S403 : BondFractions, K - O bond frac.
S404 : BondFractions, K - P bond frac.
S405 : BondFractions, K - S bond frac.
S406 : BondFractions, K - Sc bond frac.
S407 : BondFractions, K - Si bond frac.
S408 : BondFractions, K - Ti bond frac.
S409 : BondFractions, K - V bond frac.
S410 : BondFractions, K - Zn bond frac.
S411 : BondFractions, Li - Li bond frac.
S412 : BondFractions, Li - Mg bond frac.
S413 : BondFractions, Li - Mn bond frac.
S414 : BondFractions, Li - N bond frac.
S415 : BondFractions, Li - Na bond frac.
S416 : BondFractions, Li - Ni bond frac.
S417 : BondFractions, Li - O bond frac.
S418 : BondFractions, Li - P bond frac.
S419 : BondFractions, Li - S bond frac.
S420 : BondFractions, Li - Sc bond frac.
S421 : BondFractions, Li - Si bond frac.
S422 : BondFractions, Li - Ti bond frac.
S423 : BondFractions, Li - V bond frac.
S424 : BondFractions, Li - Zn bond frac.
S425 : BondFractions, Mg - Mg bond frac.
S426 : BondFractions, Mg - Mn bond frac.
S427 : BondFractions, Mg - N bond frac.
S428 : BondFractions, Mg - Na bond frac.
S429 : BondFractions, Mg - Ni bond frac.
S430 : BondFractions, Mg - O bond frac.
S431 : BondFractions, Mg - P bond frac.
S432 : BondFractions, Mg - S bond frac.
S433 : BondFractions, Mg - Sc bond frac.
S434 : BondFractions, Mg - Si bond frac.
S435 : BondFractions, Mg - Ti bond frac.
S436 : BondFractions, Mg - V bond frac.
S437 : BondFractions, Mg - Zn bond frac.
S438 : BondFractions, Mn - Mn bond frac.

S439 : BondFractions, Mn - N bond frac.
S440 : BondFractions, Mn - Na bond frac.
S441 : BondFractions, Mn - Ni bond frac.
S442 : BondFractions, Mn - O bond frac.
S443 : BondFractions, Mn - P bond frac.
S444 : BondFractions, Mn - S bond frac.
S445 : BondFractions, Mn - Sc bond frac.
S446 : BondFractions, Mn - Si bond frac.
S447 : BondFractions, Mn - Ti bond frac.
S448 : BondFractions, Mn - V bond frac.
S449 : BondFractions, Mn - Zn bond frac.
S450 : BondFractions, N - N bond frac.
S451 : BondFractions, N - Na bond frac.
S452 : BondFractions, N - Ni bond frac.
S453 : BondFractions, N - O bond frac.
S454 : BondFractions, N - P bond frac.
S455 : BondFractions, N - S bond frac.
S456 : BondFractions, N - Sc bond frac.
S457 : BondFractions, N - Si bond frac.
S458 : BondFractions, N - Ti bond frac.
S459 : BondFractions, N - V bond frac.
S460 : BondFractions, N - Zn bond frac.
S461 : BondFractions, Na - Na bond frac.
S462 : BondFractions, Na - Ni bond frac.
S463 : BondFractions, Na - O bond frac.
S464 : BondFractions, Na - P bond frac.
S465 : BondFractions, Na - S bond frac.
S466 : BondFractions, Na - Sc bond frac.
S467 : BondFractions, Na - Si bond frac.
S468 : BondFractions, Na - Ti bond frac.
S469 : BondFractions, Na - V bond frac.
S470 : BondFractions, Na - Zn bond frac.
S471 : BondFractions, Ne - Ne bond frac.
S472 : BondFractions, Ni - Ni bond frac.
S473 : BondFractions, Ni - O bond frac.
S474 : BondFractions, Ni - P bond frac.
S475 : BondFractions, Ni - S bond frac.
S476 : BondFractions, Ni - Sc bond frac.
S477 : BondFractions, Ni - Si bond frac.
S478 : BondFractions, Ni - Ti bond frac.
S479 : BondFractions, Ni - V bond frac.
S480 : BondFractions, Ni - Zn bond frac.
S481 : BondFractions, O - O bond frac.
S482 : BondFractions, O - P bond frac.
S483 : BondFractions, O - S bond frac.
S484 : BondFractions, O - Sc bond frac.
S485 : BondFractions, O - Si bond frac.
S486 : BondFractions, O - Ti bond frac.

S487 : BondFractions, O - V bond frac.
S488 : BondFractions, O - Zn bond frac.
S489 : BondFractions, P - P bond frac.
S490 : BondFractions, P - S bond frac.
S491 : BondFractions, P - Sc bond frac.
S492 : BondFractions, P - Si bond frac.
S493 : BondFractions, P - Ti bond frac.
S494 : BondFractions, P - V bond frac.
S495 : BondFractions, P - Zn bond frac.
S496 : BondFractions, S - S bond frac.
S497 : BondFractions, S - Sc bond frac.
S498 : BondFractions, S - Si bond frac.
S499 : BondFractions, S - Ti bond frac.
S500 : BondFractions, S - V bond frac.
S501 : BondFractions, S - Zn bond frac.
S502 : BondFractions, Sc - Sc bond frac.
S503 : BondFractions, Sc - Si bond frac.
S504 : BondFractions, Sc - Ti bond frac.
S505 : BondFractions, Sc - V bond frac.
S506 : BondFractions, Sc - Zn bond frac.
S507 : BondFractions, Si - Si bond frac.
S508 : BondFractions, Si - Ti bond frac.
S509 : BondFractions, Si - V bond frac.
S510 : BondFractions, Si - Zn bond frac.
S511 : BondFractions, Ti - Ti bond frac.
S512 : BondFractions, Ti - V bond frac.
S513 : BondFractions, Ti - Zn bond frac.
S514 : BondFractions, V - V bond frac.
S515 : BondFractions, V - Zn bond frac.
S516 : BondFractions, Zn - Zn bond frac.

D.3 Site

L000 : Input data, e_hull
L001 : Input data, gap pbe
L002 : Input data, mu_b
L003 : Input data, elastic anisotropy
L004 : Input data, bulk modulus
L005 : Input data, shear modulus
L006 : Input data, e_form
L007 : AGNIFingerPrint, mean AGNI eta=8.00e-01
L008 : AGNIFingerPrint, std_dev AGNI eta=8.00e-01
L009 : AGNIFingerPrint, mean AGNI eta=1.23e+00
L010 : AGNIFingerPrint, std_dev AGNI eta=1.23e+00
L011 : AGNIFingerPrint, mean AGNI eta=1.88e+00
L012 : AGNIFingerPrint, std_dev AGNI eta=1.88e+00
L013 : AGNIFingerPrint, mean AGNI eta=2.89e+00

L014 : AGNIFingerPrint, std_dev AGNI eta=2.89e+00
L015 : AGNIFingerPrint, mean AGNI eta=4.43e+00
L016 : AGNIFingerPrint, std_dev AGNI eta=4.43e+00
L017 : AGNIFingerPrint, mean AGNI eta=6.80e+00
L018 : AGNIFingerPrint, std_dev AGNI eta=6.80e+00
L019 : AGNIFingerPrint, mean AGNI eta=1.04e+01
L020 : AGNIFingerPrint, std_dev AGNI eta=1.04e+01
L021 : AGNIFingerPrint, mean AGNI eta=1.60e+01
L022 : AGNIFingerPrint, std_dev AGNI eta=1.60e+01
L023 : AGNIFingerPrint, mean AGNI dir=x eta=8.00e-01
L024 : AGNIFingerPrint, std_dev AGNI dir=x eta=8.00e-01
L025 : AGNIFingerPrint, mean AGNI dir=x eta=1.23e+00
L026 : AGNIFingerPrint, std_dev AGNI dir=x eta=1.23e+00
L027 : AGNIFingerPrint, mean AGNI dir=x eta=1.88e+00
L028 : AGNIFingerPrint, std_dev AGNI dir=x eta=1.88e+00
L029 : AGNIFingerPrint, mean AGNI dir=x eta=2.89e+00
L030 : AGNIFingerPrint, std_dev AGNI dir=x eta=2.89e+00
L031 : AGNIFingerPrint, mean AGNI dir=x eta=4.43e+00
L032 : AGNIFingerPrint, std_dev AGNI dir=x eta=4.43e+00
L033 : AGNIFingerPrint, mean AGNI dir=x eta=6.80e+00
L034 : AGNIFingerPrint, std_dev AGNI dir=x eta=6.80e+00
L035 : AGNIFingerPrint, mean AGNI dir=x eta=1.04e+01
L036 : AGNIFingerPrint, std_dev AGNI dir=x eta=1.04e+01
L037 : AGNIFingerPrint, mean AGNI dir=x eta=1.60e+01
L038 : AGNIFingerPrint, std_dev AGNI dir=x eta=1.60e+01
L039 : AGNIFingerPrint, mean AGNI dir=y eta=8.00e-01
L040 : AGNIFingerPrint, std_dev AGNI dir=y eta=8.00e-01
L041 : AGNIFingerPrint, mean AGNI dir=y eta=1.23e+00
L042 : AGNIFingerPrint, std_dev AGNI dir=y eta=1.23e+00
L043 : AGNIFingerPrint, mean AGNI dir=y eta=1.88e+00
L044 : AGNIFingerPrint, std_dev AGNI dir=y eta=1.88e+00
L045 : AGNIFingerPrint, mean AGNI dir=y eta=2.89e+00
L046 : AGNIFingerPrint, std_dev AGNI dir=y eta=2.89e+00
L047 : AGNIFingerPrint, mean AGNI dir=y eta=4.43e+00
L048 : AGNIFingerPrint, std_dev AGNI dir=y eta=4.43e+00
L049 : AGNIFingerPrint, mean AGNI dir=y eta=6.80e+00
L050 : AGNIFingerPrint, std_dev AGNI dir=y eta=6.80e+00
L051 : AGNIFingerPrint, mean AGNI dir=y eta=1.04e+01
L052 : AGNIFingerPrint, std_dev AGNI dir=y eta=1.04e+01
L053 : AGNIFingerPrint, mean AGNI dir=y eta=1.60e+01
L054 : AGNIFingerPrint, std_dev AGNI dir=y eta=1.60e+01
L055 : AGNIFingerPrint, mean AGNI dir=z eta=8.00e-01
L056 : AGNIFingerPrint, std_dev AGNI dir=z eta=8.00e-01
L057 : AGNIFingerPrint, mean AGNI dir=z eta=1.23e+00
L058 : AGNIFingerPrint, std_dev AGNI dir=z eta=1.23e+00
L059 : AGNIFingerPrint, mean AGNI dir=z eta=1.88e+00
L060 : AGNIFingerPrint, std_dev AGNI dir=z eta=1.88e+00
L061 : AGNIFingerPrint, mean AGNI dir=z eta=2.89e+00

L062 : AGNIFingerPrint, std_dev AGNI dir=z eta=2.89e+00
L063 : AGNIFingerPrint, mean AGNI dir=z eta=4.43e+00
L064 : AGNIFingerPrint, std_dev AGNI dir=z eta=4.43e+00
L065 : AGNIFingerPrint, mean AGNI dir=z eta=6.80e+00
L066 : AGNIFingerPrint, std_dev AGNI dir=z eta=6.80e+00
L067 : AGNIFingerPrint, mean AGNI dir=z eta=1.04e+01
L068 : AGNIFingerPrint, std_dev AGNI dir=z eta=1.04e+01
L069 : AGNIFingerPrint, mean AGNI dir=z eta=1.60e+01
L070 : AGNIFingerPrint, std_dev AGNI dir=z eta=1.60e+01
L071 : OPSiteFingerprint, mean sgl_bd CN_1
L072 : OPSiteFingerprint, std_dev sgl_bd CN_1
L073 : OPSiteFingerprint, mean L-shaped CN_2
L074 : OPSiteFingerprint, std_dev L-shaped CN_2
L075 : OPSiteFingerprint, mean water-like CN_2
L076 : OPSiteFingerprint, std_dev water-like CN_2
L077 : OPSiteFingerprint, mean bent 120 degrees CN_2
L078 : OPSiteFingerprint, std_dev bent 120 degrees CN_2
L079 : OPSiteFingerprint, mean bent 150 degrees CN_2
L080 : OPSiteFingerprint, std_dev bent 150 degrees CN_2
L081 : OPSiteFingerprint, mean linear CN_2
L082 : OPSiteFingerprint, std_dev linear CN_2
L083 : OPSiteFingerprint, mean trigonal planar CN_3
L084 : OPSiteFingerprint, std_dev trigonal planar CN_3
L085 : OPSiteFingerprint, mean trigonal non-coplanar CN_3
L086 : OPSiteFingerprint, std_dev trigonal non-coplanar CN_3
L087 : OPSiteFingerprint, mean T-shaped CN_3
L088 : OPSiteFingerprint, std_dev T-shaped CN_3
L089 : OPSiteFingerprint, mean square co-planar CN_4
L090 : OPSiteFingerprint, std_dev square co-planar CN_4
L091 : OPSiteFingerprint, mean tetrahedral CN_4
L092 : OPSiteFingerprint, std_dev tetrahedral CN_4
L093 : OPSiteFingerprint, mean rectangular see-saw-like CN_4
L094 : OPSiteFingerprint, std_dev rectangular see-saw-like CN_4
L095 : OPSiteFingerprint, mean see-saw-like CN_4
L096 : OPSiteFingerprint, std_dev see-saw-like CN_4
L097 : OPSiteFingerprint, mean trigonal pyramidal CN_4
L098 : OPSiteFingerprint, std_dev trigonal pyramidal CN_4
L099 : OPSiteFingerprint, mean pentagonal planar CN_5
L100 : OPSiteFingerprint, std_dev pentagonal planar CN_5
L101 : OPSiteFingerprint, mean square pyramidal CN_5
L102 : OPSiteFingerprint, std_dev square pyramidal CN_5
L103 : OPSiteFingerprint, mean trigonal bipyramidal CN_5
L104 : OPSiteFingerprint, std_dev trigonal bipyramidal CN_5
L105 : OPSiteFingerprint, mean hexagonal planar CN_6
L106 : OPSiteFingerprint, std_dev hexagonal planar CN_6
L107 : OPSiteFingerprint, mean octahedral CN_6
L108 : OPSiteFingerprint, std_dev octahedral CN_6
L109 : OPSiteFingerprint, mean pentagonal pyramidal CN_6

L110 : OPSiteFingerprint, std_dev pentagonal pyramidal CN_6
L111 : OPSiteFingerprint, mean hexagonal pyramidal CN_7
L112 : OPSiteFingerprint, std_dev hexagonal pyramidal CN_7
L113 : OPSiteFingerprint, mean pentagonal bipyramidal CN_7
L114 : OPSiteFingerprint, std_dev pentagonal bipyramidal CN_7
L115 : OPSiteFingerprint, mean body-centered cubic CN_8
L116 : OPSiteFingerprint, std_dev body-centered cubic CN_8
L117 : OPSiteFingerprint, mean hexagonal bipyramidal CN_8
L118 : OPSiteFingerprint, std_dev hexagonal bipyramidal CN_8
L119 : OPSiteFingerprint, mean q2 CN_9
L120 : OPSiteFingerprint, std_dev q2 CN_9
L121 : OPSiteFingerprint, mean q4 CN_9
L122 : OPSiteFingerprint, std_dev q4 CN_9
L123 : OPSiteFingerprint, mean q6 CN_9
L124 : OPSiteFingerprint, std_dev q6 CN_9
L125 : OPSiteFingerprint, mean q2 CN_10
L126 : OPSiteFingerprint, std_dev q2 CN_10
L127 : OPSiteFingerprint, mean q4 CN_10
L128 : OPSiteFingerprint, std_dev q4 CN_10
L129 : OPSiteFingerprint, mean q6 CN_10
L130 : OPSiteFingerprint, std_dev q6 CN_10
L131 : OPSiteFingerprint, mean q2 CN_11
L132 : OPSiteFingerprint, std_dev q2 CN_11
L133 : OPSiteFingerprint, mean q4 CN_11
L134 : OPSiteFingerprint, std_dev q4 CN_11
L135 : OPSiteFingerprint, mean q6 CN_11
L136 : OPSiteFingerprint, std_dev q6 CN_11
L137 : OPSiteFingerprint, mean cuboctahedral CN_12
L138 : OPSiteFingerprint, std_dev cuboctahedral CN_12
L139 : OPSiteFingerprint, mean q2 CN_12
L140 : OPSiteFingerprint, std_dev q2 CN_12
L141 : OPSiteFingerprint, mean q4 CN_12
L142 : OPSiteFingerprint, std_dev q4 CN_12
L143 : OPSiteFingerprint, mean q6 CN_12
L144 : OPSiteFingerprint, std_dev q6 CN_12
L145 : CrystalNNFingerprint, mean wt CN_1
L146 : CrystalNNFingerprint, std_dev wt CN_1
L147 : CrystalNNFingerprint, mean sgl_bd CN_1
L148 : CrystalNNFingerprint, std_dev sgl_bd CN_1
L149 : CrystalNNFingerprint, mean wt CN_2
L150 : CrystalNNFingerprint, std_dev wt CN_2
L151 : CrystalNNFingerprint, mean L-shaped CN_2
L152 : CrystalNNFingerprint, std_dev L-shaped CN_2
L153 : CrystalNNFingerprint, mean water-like CN_2
L154 : CrystalNNFingerprint, std_dev water-like CN_2
L155 : CrystalNNFingerprint, mean bent 120 degrees CN_2
L156 : CrystalNNFingerprint, std_dev bent 120 degrees CN_2
L157 : CrystalNNFingerprint, mean bent 150 degrees CN_2

L158 : CrystalNNFingerprint, std_dev bent 150 degrees CN_2
L159 : CrystalNNFingerprint, mean linear CN_2
L160 : CrystalNNFingerprint, std_dev linear CN_2
L161 : CrystalNNFingerprint, mean wt CN_3
L162 : CrystalNNFingerprint, std_dev wt CN_3
L163 : CrystalNNFingerprint, mean trigonal planar CN_3
L164 : CrystalNNFingerprint, std_dev trigonal planar CN_3
L165 : CrystalNNFingerprint, mean trigonal non-coplanar CN_3
L166 : CrystalNNFingerprint, std_dev trigonal non-coplanar CN_3
L167 : CrystalNNFingerprint, mean T-shaped CN_3
L168 : CrystalNNFingerprint, std_dev T-shaped CN_3
L169 : CrystalNNFingerprint, mean wt CN_4
L170 : CrystalNNFingerprint, std_dev wt CN_4
L171 : CrystalNNFingerprint, mean square co-planar CN_4
L172 : CrystalNNFingerprint, std_dev square co-planar CN_4
L173 : CrystalNNFingerprint, mean tetrahedral CN_4
L174 : CrystalNNFingerprint, std_dev tetrahedral CN_4
L175 : CrystalNNFingerprint, mean rectangular see-saw-like CN_4
L176 : CrystalNNFingerprint, std_dev rectangular see-saw-like CN_4
L177 : CrystalNNFingerprint, mean see-saw-like CN_4
L178 : CrystalNNFingerprint, std_dev see-saw-like CN_4
L179 : CrystalNNFingerprint, mean trigonal pyramidal CN_4
L180 : CrystalNNFingerprint, std_dev trigonal pyramidal CN_4
L181 : CrystalNNFingerprint, mean wt CN_5
L182 : CrystalNNFingerprint, std_dev wt CN_5
L183 : CrystalNNFingerprint, mean pentagonal planar CN_5
L184 : CrystalNNFingerprint, std_dev pentagonal planar CN_5
L185 : CrystalNNFingerprint, mean square pyramidal CN_5
L186 : CrystalNNFingerprint, std_dev square pyramidal CN_5
L187 : CrystalNNFingerprint, mean trigonal bipyramidal CN_5
L188 : CrystalNNFingerprint, std_dev trigonal bipyramidal CN_5
L189 : CrystalNNFingerprint, mean wt CN_6
L190 : CrystalNNFingerprint, std_dev wt CN_6
L191 : CrystalNNFingerprint, mean hexagonal planar CN_6
L192 : CrystalNNFingerprint, std_dev hexagonal planar CN_6
L193 : CrystalNNFingerprint, mean octahedral CN_6
L194 : CrystalNNFingerprint, std_dev octahedral CN_6
L195 : CrystalNNFingerprint, mean pentagonal pyramidal CN_6
L196 : CrystalNNFingerprint, std_dev pentagonal pyramidal CN_6
L197 : CrystalNNFingerprint, mean wt CN_7
L198 : CrystalNNFingerprint, std_dev wt CN_7
L199 : CrystalNNFingerprint, mean hexagonal pyramidal CN_7
L200 : CrystalNNFingerprint, std_dev hexagonal pyramidal CN_7
L201 : CrystalNNFingerprint, mean pentagonal bipyramidal CN_7
L202 : CrystalNNFingerprint, std_dev pentagonal bipyramidal CN_7
L203 : CrystalNNFingerprint, mean wt CN_8
L204 : CrystalNNFingerprint, std_dev wt CN_8
L205 : CrystalNNFingerprint, mean body-centered cubic CN_8

L206 : CrystalNNFingerprint, std_dev body-centered cubic CN_8
L207 : CrystalNNFingerprint, mean hexagonal bipyramidal CN_8
L208 : CrystalNNFingerprint, std_dev hexagonal bipyramidal CN_8
L209 : CrystalNNFingerprint, mean wt CN_9
L210 : CrystalNNFingerprint, std_dev wt CN_9
L211 : CrystalNNFingerprint, mean q2 CN_9
L212 : CrystalNNFingerprint, std_dev q2 CN_9
L213 : CrystalNNFingerprint, mean q4 CN_9
L214 : CrystalNNFingerprint, std_dev q4 CN_9
L215 : CrystalNNFingerprint, mean q6 CN_9
L216 : CrystalNNFingerprint, std_dev q6 CN_9
L217 : CrystalNNFingerprint, mean wt CN_10
L218 : CrystalNNFingerprint, std_dev wt CN_10
L219 : CrystalNNFingerprint, mean q2 CN_10
L220 : CrystalNNFingerprint, std_dev q2 CN_10
L221 : CrystalNNFingerprint, mean q4 CN_10
L222 : CrystalNNFingerprint, std_dev q4 CN_10
L223 : CrystalNNFingerprint, mean q6 CN_10
L224 : CrystalNNFingerprint, std_dev q6 CN_10
L225 : CrystalNNFingerprint, mean wt CN_11
L226 : CrystalNNFingerprint, std_dev wt CN_11
L227 : CrystalNNFingerprint, mean q2 CN_11
L228 : CrystalNNFingerprint, std_dev q2 CN_11
L229 : CrystalNNFingerprint, mean q4 CN_11
L230 : CrystalNNFingerprint, std_dev q4 CN_11
L231 : CrystalNNFingerprint, mean q6 CN_11
L232 : CrystalNNFingerprint, std_dev q6 CN_11
L233 : CrystalNNFingerprint, mean wt CN_12
L234 : CrystalNNFingerprint, std_dev wt CN_12
L235 : CrystalNNFingerprint, mean cuboctahedral CN_12
L236 : CrystalNNFingerprint, std_dev cuboctahedral CN_12
L237 : CrystalNNFingerprint, mean q2 CN_12
L238 : CrystalNNFingerprint, std_dev q2 CN_12
L239 : CrystalNNFingerprint, mean q4 CN_12
L240 : CrystalNNFingerprint, std_dev q4 CN_12
L241 : CrystalNNFingerprint, mean q6 CN_12
L242 : CrystalNNFingerprint, std_dev q6 CN_12
L243 : CrystalNNFingerprint, mean wt CN_13
L244 : CrystalNNFingerprint, std_dev wt CN_13
L245 : CrystalNNFingerprint, mean wt CN_14
L246 : CrystalNNFingerprint, std_dev wt CN_14
L247 : CrystalNNFingerprint, mean wt CN_15
L248 : CrystalNNFingerprint, std_dev wt CN_15
L249 : CrystalNNFingerprint, mean wt CN_16
L250 : CrystalNNFingerprint, std_dev wt CN_16
L251 : CrystalNNFingerprint, mean wt CN_17
L252 : CrystalNNFingerprint, std_dev wt CN_17
L253 : CrystalNNFingerprint, mean wt CN_18

L254 : CrystalNNFingerprint, std_dev wt CN_18
L255 : CrystalNNFingerprint, mean wt CN_19
L256 : CrystalNNFingerprint, std_dev wt CN_19
L257 : CrystalNNFingerprint, mean wt CN_20
L258 : CrystalNNFingerprint, std_dev wt CN_20
L259 : CrystalNNFingerprint, mean wt CN_21
L260 : CrystalNNFingerprint, std_dev wt CN_21
L261 : CrystalNNFingerprint, mean wt CN_22
L262 : CrystalNNFingerprint, std_dev wt CN_22
L263 : CrystalNNFingerprint, mean wt CN_23
L264 : CrystalNNFingerprint, std_dev wt CN_23
L265 : CrystalNNFingerprint, mean wt CN_24
L266 : CrystalNNFingerprint, std_dev wt CN_24
L267 : VoronoiFingerprint, mean Voro_index_3
L268 : VoronoiFingerprint, std_dev Voro_index_3
L269 : VoronoiFingerprint, mean Voro_index_4
L270 : VoronoiFingerprint, std_dev Voro_index_4
L271 : VoronoiFingerprint, mean Voro_index_5
L272 : VoronoiFingerprint, std_dev Voro_index_5
L273 : VoronoiFingerprint, mean Voro_index_6
L274 : VoronoiFingerprint, std_dev Voro_index_6
L275 : VoronoiFingerprint, mean Voro_index_7
L276 : VoronoiFingerprint, std_dev Voro_index_7
L277 : VoronoiFingerprint, mean Voro_index_8
L278 : VoronoiFingerprint, std_dev Voro_index_8
L279 : VoronoiFingerprint, mean Voro_index_9
L280 : VoronoiFingerprint, std_dev Voro_index_9
L281 : VoronoiFingerprint, mean Voro_index_10
L282 : VoronoiFingerprint, std_dev Voro_index_10
L283 : VoronoiFingerprint, mean Symmetry_index_3
L284 : VoronoiFingerprint, std_dev Symmetry_index_3
L285 : VoronoiFingerprint, mean Symmetry_index_4
L286 : VoronoiFingerprint, std_dev Symmetry_index_4
L287 : VoronoiFingerprint, mean Symmetry_index_5
L288 : VoronoiFingerprint, std_dev Symmetry_index_5
L289 : VoronoiFingerprint, mean Symmetry_index_6
L290 : VoronoiFingerprint, std_dev Symmetry_index_6
L291 : VoronoiFingerprint, mean Symmetry_index_7
L292 : VoronoiFingerprint, std_dev Symmetry_index_7
L293 : VoronoiFingerprint, mean Symmetry_index_8
L294 : VoronoiFingerprint, std_dev Symmetry_index_8
L295 : VoronoiFingerprint, mean Symmetry_index_9
L296 : VoronoiFingerprint, std_dev Symmetry_index_9
L297 : VoronoiFingerprint, mean Symmetry_index_10
L298 : VoronoiFingerprint, std_dev Symmetry_index_10
L299 : VoronoiFingerprint, mean Voro_vol_sum
L300 : VoronoiFingerprint, std_dev Voro_vol_sum
L301 : VoronoiFingerprint, mean Voro_area_sum

L302 : VoronoiFingerprint, std_dev Voro_area_sum
L303 : VoronoiFingerprint, mean Voro_vol_mean
L304 : VoronoiFingerprint, std_dev Voro_vol_mean
L305 : VoronoiFingerprint, mean Voro_vol_std_dev
L306 : VoronoiFingerprint, std_dev Voro_vol_std_dev
L307 : VoronoiFingerprint, mean Voro_vol_minimum
L308 : VoronoiFingerprint, std_dev Voro_vol_minimum
L309 : VoronoiFingerprint, mean Voro_vol_maximum
L310 : VoronoiFingerprint, std_dev Voro_vol_maximum
L311 : VoronoiFingerprint, mean Voro_area_mean
L312 : VoronoiFingerprint, std_dev Voro_area_mean
L313 : VoronoiFingerprint, mean Voro_area_std_dev
L314 : VoronoiFingerprint, std_dev Voro_area_std_dev
L315 : VoronoiFingerprint, mean Voro_area_minimum
L316 : VoronoiFingerprint, std_dev Voro_area_minimum
L317 : VoronoiFingerprint, mean Voro_area_maximum
L318 : VoronoiFingerprint, std_dev Voro_area_maximum
L319 : VoronoiFingerprint, mean Voro_dist_mean
L320 : VoronoiFingerprint, std_dev Voro_dist_mean
L321 : VoronoiFingerprint, mean Voro_dist_std_dev
L322 : VoronoiFingerprint, std_dev Voro_dist_std_dev
L323 : VoronoiFingerprint, mean Voro_dist_minimum
L324 : VoronoiFingerprint, std_dev Voro_dist_minimum
L325 : VoronoiFingerprint, mean Voro_dist_maximum
L326 : VoronoiFingerprint, std_dev Voro_dist_maximum
L327 : GaussianSymmFunc, mean G2_0.05
L328 : GaussianSymmFunc, std_dev G2_0.05
L329 : GaussianSymmFunc, mean G2_4.0
L330 : GaussianSymmFunc, std_dev G2_4.0
L331 : GaussianSymmFunc, mean G2_20.0
L332 : GaussianSymmFunc, std_dev G2_20.0
L333 : GaussianSymmFunc, mean G2_80.0
L334 : GaussianSymmFunc, std_dev G2_80.0
L335 : GaussianSymmFunc, mean G4_0.005_1.0_1.0
L336 : GaussianSymmFunc, std_dev G4_0.005_1.0_1.0
L337 : GaussianSymmFunc, mean G4_0.005_1.0_-1.0
L338 : GaussianSymmFunc, std_dev G4_0.005_1.0_-1.0
L339 : GaussianSymmFunc, mean G4_0.005_4.0_1.0
L340 : GaussianSymmFunc, std_dev G4_0.005_4.0_1.0
L341 : GaussianSymmFunc, mean G4_0.005_4.0_-1.0
L342 : GaussianSymmFunc, std_dev G4_0.005_4.0_-1.0
L343 : ChemEnvSiteFingerprint, mean S:1
L344 : ChemEnvSiteFingerprint, std_dev S:1
L345 : ChemEnvSiteFingerprint, mean L:2
L346 : ChemEnvSiteFingerprint, std_dev L:2
L347 : ChemEnvSiteFingerprint, mean A:2
L348 : ChemEnvSiteFingerprint, std_dev A:2
L349 : ChemEnvSiteFingerprint, mean TL:3

L350 : ChemEnvSiteFingerprint, std_dev TL:3
L351 : ChemEnvSiteFingerprint, mean TY:3
L352 : ChemEnvSiteFingerprint, std_dev TY:3
L353 : ChemEnvSiteFingerprint, mean TS:3
L354 : ChemEnvSiteFingerprint, std_dev TS:3
L355 : ChemEnvSiteFingerprint, mean T:4
L356 : ChemEnvSiteFingerprint, std_dev T:4
L357 : ChemEnvSiteFingerprint, mean S:4
L358 : ChemEnvSiteFingerprint, std_dev S:4
L359 : ChemEnvSiteFingerprint, mean SY:4
L360 : ChemEnvSiteFingerprint, std_dev SY:4
L361 : ChemEnvSiteFingerprint, mean SS:4
L362 : ChemEnvSiteFingerprint, std_dev SS:4
L363 : ChemEnvSiteFingerprint, mean PP:5
L364 : ChemEnvSiteFingerprint, std_dev PP:5
L365 : ChemEnvSiteFingerprint, mean S:5
L366 : ChemEnvSiteFingerprint, std_dev S:5
L367 : ChemEnvSiteFingerprint, mean T:5
L368 : ChemEnvSiteFingerprint, std_dev T:5
L369 : ChemEnvSiteFingerprint, mean O:6
L370 : ChemEnvSiteFingerprint, std_dev O:6
L371 : ChemEnvSiteFingerprint, mean T:6
L372 : ChemEnvSiteFingerprint, std_dev T:6
L373 : ChemEnvSiteFingerprint, mean PP:6
L374 : ChemEnvSiteFingerprint, std_dev PP:6
L375 : ChemEnvSiteFingerprint, mean PB:7
L376 : ChemEnvSiteFingerprint, std_dev PB:7
L377 : ChemEnvSiteFingerprint, mean ST:7
L378 : ChemEnvSiteFingerprint, std_dev ST:7
L379 : ChemEnvSiteFingerprint, mean ET:7
L380 : ChemEnvSiteFingerprint, std_dev ET:7
L381 : ChemEnvSiteFingerprint, mean FO:7
L382 : ChemEnvSiteFingerprint, std_dev FO:7
L383 : ChemEnvSiteFingerprint, mean C:8
L384 : ChemEnvSiteFingerprint, std_dev C:8
L385 : ChemEnvSiteFingerprint, mean SA:8
L386 : ChemEnvSiteFingerprint, std_dev SA:8
L387 : ChemEnvSiteFingerprint, mean SBT:8
L388 : ChemEnvSiteFingerprint, std_dev SBT:8
L389 : ChemEnvSiteFingerprint, mean TBT:8
L390 : ChemEnvSiteFingerprint, std_dev TBT:8
L391 : ChemEnvSiteFingerprint, mean DD:8
L392 : ChemEnvSiteFingerprint, std_dev DD:8
L393 : ChemEnvSiteFingerprint, mean DDPN:8
L394 : ChemEnvSiteFingerprint, std_dev DDPN:8
L395 : ChemEnvSiteFingerprint, mean HB:8
L396 : ChemEnvSiteFingerprint, std_dev HB:8
L397 : ChemEnvSiteFingerprint, mean BO_1:8

L398 : ChemEnvSiteFingerprint, std_dev BO_1:8
L399 : ChemEnvSiteFingerprint, mean BO_2:8
L400 : ChemEnvSiteFingerprint, std_dev BO_2:8
L401 : ChemEnvSiteFingerprint, mean BO_3:8
L402 : ChemEnvSiteFingerprint, std_dev BO_3:8
L403 : ChemEnvSiteFingerprint, mean TC:9
L404 : ChemEnvSiteFingerprint, std_dev TC:9
L405 : ChemEnvSiteFingerprint, mean TT_1:9
L406 : ChemEnvSiteFingerprint, std_dev TT_1:9
L407 : ChemEnvSiteFingerprint, mean TT_2:9
L408 : ChemEnvSiteFingerprint, std_dev TT_2:9
L409 : ChemEnvSiteFingerprint, mean TT_3:9
L410 : ChemEnvSiteFingerprint, std_dev TT_3:9
L411 : ChemEnvSiteFingerprint, mean HD:9
L412 : ChemEnvSiteFingerprint, std_dev HD:9
L413 : ChemEnvSiteFingerprint, mean TI:9
L414 : ChemEnvSiteFingerprint, std_dev TI:9
L415 : ChemEnvSiteFingerprint, mean SMA:9
L416 : ChemEnvSiteFingerprint, std_dev SMA:9
L417 : ChemEnvSiteFingerprint, mean SS:9
L418 : ChemEnvSiteFingerprint, std_dev SS:9
L419 : ChemEnvSiteFingerprint, mean TO_1:9
L420 : ChemEnvSiteFingerprint, std_dev TO_1:9
L421 : ChemEnvSiteFingerprint, mean TO_2:9
L422 : ChemEnvSiteFingerprint, std_dev TO_2:9
L423 : ChemEnvSiteFingerprint, mean TO_3:9
L424 : ChemEnvSiteFingerprint, std_dev TO_3:9
L425 : ChemEnvSiteFingerprint, mean PP:10
L426 : ChemEnvSiteFingerprint, std_dev PP:10
L427 : ChemEnvSiteFingerprint, mean PA:10
L428 : ChemEnvSiteFingerprint, std_dev PA:10
L429 : ChemEnvSiteFingerprint, mean SBSA:10
L430 : ChemEnvSiteFingerprint, std_dev SBSA:10
L431 : ChemEnvSiteFingerprint, mean MI:10
L432 : ChemEnvSiteFingerprint, std_dev MI:10
L433 : ChemEnvSiteFingerprint, mean S:10
L434 : ChemEnvSiteFingerprint, std_dev S:10
L435 : ChemEnvSiteFingerprint, mean H:10
L436 : ChemEnvSiteFingerprint, std_dev H:10
L437 : ChemEnvSiteFingerprint, mean BS_1:10
L438 : ChemEnvSiteFingerprint, std_dev BS_1:10
L439 : ChemEnvSiteFingerprint, mean BS_2:10
L440 : ChemEnvSiteFingerprint, std_dev BS_2:10
L441 : ChemEnvSiteFingerprint, mean TBSA:10
L442 : ChemEnvSiteFingerprint, std_dev TBSA:10
L443 : ChemEnvSiteFingerprint, mean PCPA:11
L444 : ChemEnvSiteFingerprint, std_dev PCPA:11
L445 : ChemEnvSiteFingerprint, mean H:11

L446 : ChemEnvSiteFingerprint, std_dev H:11
L447 : ChemEnvSiteFingerprint, mean SH:11
L448 : ChemEnvSiteFingerprint, std_dev SH:11
L449 : ChemEnvSiteFingerprint, mean CO:11
L450 : ChemEnvSiteFingerprint, std_dev CO:11
L451 : ChemEnvSiteFingerprint, mean DI:11
L452 : ChemEnvSiteFingerprint, std_dev DI:11
L453 : ChemEnvSiteFingerprint, mean I:12
L454 : ChemEnvSiteFingerprint, std_dev I:12
L455 : ChemEnvSiteFingerprint, mean PBP:12
L456 : ChemEnvSiteFingerprint, std_dev PBP:12
L457 : ChemEnvSiteFingerprint, mean TT:12
L458 : ChemEnvSiteFingerprint, std_dev TT:12
L459 : ChemEnvSiteFingerprint, mean C:12
L460 : ChemEnvSiteFingerprint, std_dev C:12
L461 : ChemEnvSiteFingerprint, mean AC:12
L462 : ChemEnvSiteFingerprint, std_dev AC:12
L463 : ChemEnvSiteFingerprint, mean SC:12
L464 : ChemEnvSiteFingerprint, std_dev SC:12
L465 : ChemEnvSiteFingerprint, mean S:12
L466 : ChemEnvSiteFingerprint, std_dev S:12
L467 : ChemEnvSiteFingerprint, mean HP:12
L468 : ChemEnvSiteFingerprint, std_dev HP:12
L469 : ChemEnvSiteFingerprint, mean HA:12
L470 : ChemEnvSiteFingerprint, std_dev HA:12
L471 : ChemEnvSiteFingerprint, mean SH:13
L472 : ChemEnvSiteFingerprint, std_dev SH:13
L473 : ChemEnvSiteFingerprint, mean DD:20
L474 : ChemEnvSiteFingerprint, std_dev DD:20
L475 : CoordinationNumber, mean CN_VoronoiNN
L476 : CoordinationNumber, std_dev CN_VoronoiNN
L477 : GeneralizedRDF, mean Gaussian center=0.0 width=1.0
L478 : GeneralizedRDF, std_dev Gaussian center=0.0 width=1.0
L479 : GeneralizedRDF, mean Gaussian center=1.0 width=1.0
L480 : GeneralizedRDF, std_dev Gaussian center=1.0 width=1.0
L481 : GeneralizedRDF, mean Gaussian center=2.0 width=1.0
L482 : GeneralizedRDF, std_dev Gaussian center=2.0 width=1.0
L483 : GeneralizedRDF, mean Gaussian center=3.0 width=1.0
L484 : GeneralizedRDF, std_dev Gaussian center=3.0 width=1.0
L485 : GeneralizedRDF, mean Gaussian center=4.0 width=1.0
L486 : GeneralizedRDF, std_dev Gaussian center=4.0 width=1.0
L487 : GeneralizedRDF, mean Gaussian center=5.0 width=1.0
L488 : GeneralizedRDF, std_dev Gaussian center=5.0 width=1.0
L489 : GeneralizedRDF, mean Gaussian center=6.0 width=1.0
L490 : GeneralizedRDF, std_dev Gaussian center=6.0 width=1.0
L491 : GeneralizedRDF, mean Gaussian center=7.0 width=1.0
L492 : GeneralizedRDF, std_dev Gaussian center=7.0 width=1.0
L493 : GeneralizedRDF, mean Gaussian center=8.0 width=1.0

L494 : GeneralizedRDF, std_dev Gaussian center=8.0 width=1.0
L495 : GeneralizedRDF, mean Gaussian center=9.0 width=1.0
L496 : GeneralizedRDF, std_dev Gaussian center=9.0 width=1.0
L497 : LocalPropertyDifference, mean local difference in Electronegativity
L498 : LocalPropertyDifference, std_dev local difference in Electronegativity
L499 : BondOrientationParameter, mean BOOP Q l=1
L500 : BondOrientationParameter, std_dev BOOP Q l=1
L501 : BondOrientationParameter, mean BOOP Q l=2
L502 : BondOrientationParameter, std_dev BOOP Q l=2
L503 : BondOrientationParameter, mean BOOP Q l=3
L504 : BondOrientationParameter, std_dev BOOP Q l=3
L505 : BondOrientationParameter, mean BOOP Q l=4
L506 : BondOrientationParameter, std_dev BOOP Q l=4
L507 : BondOrientationParameter, mean BOOP Q l=5
L508 : BondOrientationParameter, std_dev BOOP Q l=5
L509 : BondOrientationParameter, mean BOOP Q l=6
L510 : BondOrientationParameter, std_dev BOOP Q l=6
L511 : BondOrientationParameter, mean BOOP Q l=7
L512 : BondOrientationParameter, std_dev BOOP Q l=7
L513 : BondOrientationParameter, mean BOOP Q l=8
L514 : BondOrientationParameter, std_dev BOOP Q l=8
L515 : BondOrientationParameter, mean BOOP Q l=9
L516 : BondOrientationParameter, std_dev BOOP Q l=9
L517 : BondOrientationParameter, mean BOOP Q l=10
L518 : BondOrientationParameter, std_dev BOOP Q l=10
L519 : AverageBondLength, mean Average bond length
L520 : AverageBondLength, std_dev Average bond length
L521 : AverageBondAngle, mean Average bond angle
L522 : AverageBondAngle, std_dev Average bond angle

Bibliography

[1] C. L. Magee, "Towards quantification of the role of materials innovation in overall technological development", *Complexity*, vol. 18, no. 1, pp. 10–25, Sep. 2012.

[2] G. R. Schleder, A. C. M. Padilha, C. M. Acosta, M. Costa, and A. Fazzio, "From DFT to machine learning: Recent approaches to materials science–a review", *Journal of Physics: Materials*, vol. 2, no. 3, p. 032 001, May 16, 2019.

[3] P. A. M. Dirac, "Quantum mechanics of many-electron systems", *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 123, no. 792, pp. 714–733, Apr. 6, 1929.

[4] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas", *Physical Review*, vol. 136, no. 3, B864–B871, Nov. 9, 1964.

[5] Materials project, [Online]. Available at: `https://materialsproject.org/` . Last accessed: Apr. 14, 2019.

[6] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science", *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.

[7] M. A. Alizade, R. Bressler, and K. Brendel, "Stereochemistry of the hydrogen transfer to NAD catalyzed by (s)alanine dehydrogenase from bacillus subtilis", *Biochimica Et Biophysica Acta*, vol. 397, no. 1, pp. 5–8, Jul. 27, 1975.

[8] ABINIT, [Online]. Available at: `https://www.abinit.org/` . Last accessed: Jun. 9, 2019.

[9] Aflow - automatic - FLOW for materials discovery, [Online]. Available at: `http://aflowlib.org/` . Last accessed: Jun. 9, 2019.

[10] Materials cloud, [Online]. Available at: `https://www.materialscloud.org/` . Last accessed: Jun. 9, 2019.

[11] G. Petretto, S. Dwaraknath, H. P. C. Miranda, D. Winston, M. Giantomassi, M. J. van Setten, X. Gonze, K. A. Persson, G. Hautier, and G.-M. Rignanese, "High-throughput density-functional perturbation theory phonons for inorganic materials", *Scientific Data*, vol. 5, p. 180 065, May 1, 2018.

[12] S. Raschka. Learning best practices for model evaluation and hyperparameter tuning, [Online]. Available at: `https://nbviewer.jupyter.org/github/rasbt/python-machine-learning-book/blob/master/code/ch06/ch06.ipynb#K-fold-cross-validation` . Last accessed: Apr. 2, 2019.

[13] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1993.

[14] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials", *The Journal of Chemical Physics*, vol. 134, no. 7, p. 074 106, Feb. 16, 2011.

[15] L. Bottou, "Large-scale machine learning with stochastic gradient descent", in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds., Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[17] N. Kiselyova, V. Gladun, and N. Vashchenko, "Computational materials design using artificial intelligence methods", *Journal of Alloys and Compounds*, vol. 279, no. 1, pp. 8–13, Sep. 1998.

[18] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. M. Wolverton, "Materials design and discovery with high-throughput density functional theory: The open quantum materials database (OQMD)", *JOM*, vol. 65, no. 11, pp. 1501–1509, Nov. 1, 2013.

[19] G. Pilania, C. Wang, X. Jiang, S. Rajasekaran, and R. Ramprasad, "Accelerating materials property predictions using machine learning", *Scientific Reports*, vol. 3, p. 2810, Sep. 30, 2013.

[20] G. Hautier, C. C. Fischer, A. Jain, T. Mueller, and G. Ceder, "Finding nature's missing ternary oxide compounds using machine learning and density functional theory", *Chemistry of Materials*, vol. 22, no. 12, pp. 3762–3767, Jun. 22, 2010.

[21] A. O. Oliynyk, E. Antono, T. D. Sparks, L. Ghadbeigi, M. W. Gaultois, B. Meredig, and A. Mar, "High-throughput machine-learning-driven synthesis of full-heusler compounds", *Chemistry of Materials*, vol. 28, no. 20, pp. 7324–7331, Oct. 25, 2016.

[22] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning", *Physical Review Letters*, vol. 108, no. 5, p. 058 301, Jan. 31, 2012.

[23] H. T. Stokes and D. M. Hatch, "FINDSYM : Program for identifying the space-group symmetry of a crystal", *Journal of Applied Crystallography*, vol. 38, no. 1, pp. 237–238, Feb. 1, 2005.

[24] A. Ziletti, D. Kumar, M. Scheffler, and L. M. Ghiringhelli, "Insightful classification of crystal structures using deep learning", *Nature Communications*, vol. 9, no. 1, Dec. 2018.

[25] Y. Saad, D. Gao, T. Ngo, S. Bobbitt, J. R. Chelikowsky, and W. Andreoni, "Data mining for materials: Computational experiments with a b compounds", *Physical Review B*, vol. 85, no. 10, Mar. 6, 2012.

[26] C. Zeni, K. Rossi, A. Glielmo, Á. Fekete, N. Gaston, F. Baletto, and A. De Vita, "Building machine learning force fields for nanoclusters", *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241 739, Jun. 28, 2018.

[27] B. Kolb, L. C. Lentz, and A. M. Kolpak, "Discovering charge density functionals and structure-property relationships with PROPhet: A general framework for coupling machine learning and first-principles methods", *Scientific Reports*, vol. 7, no. 1, Dec. 2017.

[28] T. Lam Pham, H. Kino, K. Terakura, T. Miyake, K. Tsuda, I. Takigawa, and H. Chi Dam, "Machine learning reveals orbital interaction in materials", *Science and Technology of Advanced Materials*, vol. 18, no. 1, pp. 756–765, Dec. 31, 2017.

[29] V. Stanev, C. Oses, A. G. Kusne, E. Rodriguez, J. Paglione, S. Curtarolo, and I. Takeuchi, "Machine learning modeling of superconducting critical temperature", *npj Computational Materials*, vol. 4, no. 1, Dec. 2018.

[30] L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton, "A general-purpose machine learning framework for predicting properties of inorganic materials", *npj Computational Materials*, vol. 2, p. 16 028, Aug. 26, 2016.

[31] Superconducting material database (SuperCon), [Online]. Available at: `https://supercon.nims.go.jp/index_en.html` . Last accessed: Jun. 1, 2019.

[32] F. Legrain, J. Carrete, A. van Roekeghem, S. Curtarolo, and N. Mingo, "How chemical composition alone can predict vibrational free energies and entropies of solids", *Chemistry of Materials*, vol. 29, no. 15, pp. 6220–6227, Aug. 8, 2017.

[33] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties", *Physical Review Letters*, vol. 120, no. 14, Apr. 6, 2018.

[34] P. Verpoort, P. MacDonald, and G. Conduit, "Materials data validation and imputation with an artificial neural network", *Computational Materials Science*, vol. 147, pp. 176–185, May 2018.

[35] OQMD, [Online]. Available at: `http://oqmd.org/` . Last accessed: Apr. 14, 2019.

[36] A. Seko, T. Maekawa, K. Tsuda, and I. Tanaka, "Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single- and binary-component solids", *Physical Review B*, vol. 89, no. 5, p. 054 303, Feb. 18, 2014.

[37] B. Meredig, A. Agrawal, S. Kirklin, J. E. Saal, J. W. Doak, A. Thompson, K. Zhang, A. Choudhary, and C. Wolverton, "Combinatorial screening for new materials in unconstrained composition space with machine learning", *Physical Review B*, vol. 89, no. 9, p. 094 104, Mar. 14, 2014.

[38] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, "Big data of materials science: Critical role of the descriptor", *Physical Review Letters*, vol. 114, no. 10, p. 105 503, Mar. 13, 2015.

[39] C. S. Kong, W. Luo, S. Arapan, P. Villars, S. Iwata, R. Ahuja, and K. Rajan, "Information-theoretic approach for the discovery of design rules for crystal chemistry", *Journal of Chemical Information and Modeling*, vol. 52, no. 7, pp. 1812–1820, Jul. 23, 2012.

[40] G. Hautier, C. Fischer, V. Ehrlacher, A. Jain, and G. Ceder, "Data mined ionic substitutions for the discovery of new compounds", *Inorganic Chemistry*, vol. 50, no. 2, pp. 656–663, Jan. 17, 2011.

[41] C. C. Fischer, K. J. Tibbetts, D. Morgan, and G. Ceder, "Predicting crystal structure by merging data mining with quantum mechanics", *Nature Materials*, vol. 5, no. 8, pp. 641–646, Aug. 2006.

[42] S. Curtarolo, D. Morgan, K. Persson, J. Rodgers, and G. Ceder, "Predicting crystal structures with data mining of quantum calculations", *Physical Review Letters*, vol. 91, no. 13, p. 135 503, Sep. 24, 2003.

[43] P. Dey, J. Bible, S. Datta, S. Broderick, J. Jasinski, M. Sunkara, M. Menon, and K. Rajan, "Informatics-aided bandgap engineering for solar materials", *Computational Materials Science*, vol. 83, pp. 185–195, Feb. 15, 2014.

[44] G. Pilania, A. Mannodi-Kanakkithodi, B. Uberuaga, R. Ramprasad, J. E. Guberntis, and T. Lookman, "Machine learning bandgaps of double perovskites", *Scientific Reports*, vol. 6, Dec. 5, 2015.

[45] S. Chatterjee, M. Murugananth, and H. K. D. H. Bhadeshia, "Delta-TRIP steel", *Materials Science and Technology*, vol. 23, no. 7, pp. 819–827, Jul. 1, 2007.

[46] H. K. D. H. Bhadeshia, R. C. Dimitriu, S. Forsik, J. H. Pak, and J. H. Ryu, "Performance of neural networks in materials science", *Materials Science and Technology*, vol. 25, no. 4, pp. 504–510, Apr. 2009.

[47] L. Ward, R. Liu, A. Krishna, V. I. Hegde, A. Agrawal, A. Choudhary, and C. Wolverton, "Including crystal structure attributes in machine learning models of formation energies via voronoi tessellations", *Physical Review B*, vol. 96, no. 2, p. 024 104, Jul. 14, 2017.

[48] L. Ward, A. Dunn, A. Faghaninia, N. E. R. Zimmermann, S. Bajaj, Q. Wang, J. Montoya, J. Chen, K. Bystrom, M. Dylla, K. Chard, M. Asta, K. A. Persson, G. J. Snyder, I. Foster, and A. Jain, "Matminer: An open source toolkit for materials data mining", *Computational Materials Science*, vol. 152, pp. 60–69, Sep. 1, 2018.

[49] Citrination, [Online]. Available at: `https://citrination.com/contributing` . Last accessed: Jun. 3, 2019.

[50] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, "Commentary: The materials project: A materials genome approach to accelerating materials innovation", *APL Materials*, vol. 1, no. 1, p. 011 002, Jul. 1, 2013.

[51] B. Blaiszik, K. Chard, J. Pruyne, R. Ananthakrishnan, S. Tuecke, and I. Foster, "The materials data facility: Data services to advance materials science research", *JOM*, vol. 68, no. 8, pp. 2045–2052, Aug. 2016.

[52] MPDS materials platform, [Online]. Available at: `https://mpds.io/#start` . Last accessed: Jun. 3, 2019.

[53] Hackingmaterials/matminer: Data mining for materials science, [Online]. Available at: `https://github.com/hackingmaterials/matminer` . Last accessed: Apr. 16, 2019.

[54] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, "Python materials genomics (pymatgen): A robust, open-source python library for materials analysis", *Computational Materials Science*, vol. 68, pp. 314–319, Feb. 1, 2013.

[55] A. M. Deml, R. O'Hayre, C. Wolverton, and V. Stevanović, "Predicting density functional theory total energies and enthalpies of formation of metal-nonmetal compounds by linear regression", *Physical Review B*, vol. 93, no. 8, Feb. 29, 2016.

[56] S. Kotochigova, Z. H. Levine, E. L. Shirley, M. D. Stiles, and C. W. Clark, "Local-density-functional calculations of the energy of atoms", *Physical Review A*, vol. 55, no. 1, pp. 191–199, Jan. 1, 1997.

[57] K. J. Laws, D. B. Miracle, and M. Ferry, "A predictive structural model for bulk metallic glasses", *Nature Communications*, vol. 6, no. 1, Dec. 2015.

[58] M. A. Butler, "Prediction of flatband potentials at semiconductor-electrolyte interfaces from atomic electronegativities", *Journal of The Electrochemical Society*, vol. 125, no. 2, p. 228, 1978.

[59] C. Kittel, *Introduction to solid state physics*, 8. ed. Hoboken, NJ: Wiley, 2005, 680 pp., OCLC: 255250235.

[60] R. F. Zhang, S. H. Zhang, Z. J. He, J. Jing, and S. H. Sheng, "Miedema calculator: A thermodynamic platform for predicting formation enthalpies of alloys within framework of miedema's theory", *Computer Physics Communications*, vol. C, no. 209, pp. 58–69, 2016.

[61] X. Yang and Y. Zhang, "Prediction of high-entropy stabilized solid-solution in multi-component alloys", *Materials Chemistry and Physics*, vol. 132, no. 2, pp. 233–238, Feb. 2012.

[62] D. B. Miracle, D. V. Louzguine-Luzgin, L. V. Louzguina-Luzgina, and A. Inoue, "An assessment of binary metallic glasses: Correlations between structure, glass forming ability and stability", *International Materials Reviews*, vol. 55, no. 4, pp. 218–256, Jul. 1, 2010.

[63] P. P. Ewald, "Die Berechnung optischer und elektrostatischer Gitterpotentiale", *Annalen der Physik*, vol. 369, no. 3, pp. 253–287, 1921.

[64] Radial distribution function - wikipedia, [Online]. Available at: `https://en.wikipedia.org/w/index.php?title=Radial_distribution_function&oldid=887892627` . Last accessed: Jun. 4, 2019.

[65] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, "Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space", *The Journal of Physical Chemistry Letters*, vol. 6, no. 12, pp. 2326–2331, Jun. 18, 2015.

[66] F. Faber, A. Lindmaa, O. A. v. Lilienfeld, and R. Armiento, "Crystal structure representations for machine learning models of formation energies", *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1094–1101, 2015.

[67] M. Sprik, "Coordination numbers as reaction coordinates in constrained molecular dynamics", *Faraday Discussions*, vol. 110, no. 0, pp. 437–445, Jan. 1, 1998.

[68] N. E. R. Zimmermann, M. K. Horton, A. Jain, and M. Haranczyk, "Assessing local structure motifs using order parameters for motif recognition, interstitial identification, and diffusion path characterization", *Frontiers in Materials*, vol. 4, Nov. 13, 2017.

[69] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2 edition. Chichester ; New York: Wiley, Jul. 26, 2000, 696 pp.

[70] Voronoi diagram - wikipedia, [Online]. Available at: `https://en.wikipedia.org/wiki/Voronoi_diagram` . Last accessed: Jun. 3, 2019.

[71] S. Falco, J. Jiang, F. De Cola, and N. Petrinic, "Generation of 3d polycrystalline microstructures with a conditioned laguerre-voronoi tessellation technique", *Computational Materials Science*, vol. 136, pp. 20–28, Aug. 2017.

[72] B. Peters, "Competing nucleation pathways in a mixture of oppositely charged colloids: Out-of-equilibrium nucleation revisited", *The Journal of Chemical Physics*, vol. 131, no. 24, p. 244 103, Dec. 28, 2009.

[73] P. J. Steinhardt, D. R. Nelson, and M. Ronchetti, "Bond-orientational order in liquids and glasses", *Physical Review B*, vol. 28, no. 2, pp. 784–805, Jul. 15, 1983.

[74] A. Khorshidi and A. A. Peterson, "Amp: A modular approach to machine learning in atomistic simulations", *Computer Physics Communications*, vol. 207, pp. 310–324, Oct. 1, 2016.

[75] D. Waroquiers, X. Gonze, G.-M. Rignanese, C. Welker-Nieuwoudt, F. Rosowski, M. Göbel, S. Schenk, P. Degelmann, R. André, R. Glaum, and G. Hautier, "Statistical analysis of coordination environments in oxides", *Chemistry of Materials*, vol. 29, no. 19, pp. 8346–8360, Oct. 10, 2017.

[76] M. O'Keeffe, "A proposed rigorous definition of coordination number", *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 35, no. 5, pp. 772–775, Sep. 1, 1979.

[77] M. Pinsky and D. Avnir, "Continuous symmetry measures. 5. the classical polyhedra", *Inorganic Chemistry*, vol. 37, no. 21, pp. 5575–5582, Oct. 1, 1998.

[78] V. Botu and R. Ramprasad, "Adaptive machine learning framework to accelerate ab initio molecular dynamics", *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1074–1083, 2015.

[79] G. Voronoi, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs.", *Journal für die reine und angewandte Mathematik*, vol. 134, pp. 198–287, 1908.

[80] G. L. Dirichlet, "Über die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen.", *Journal für die reine und angewandte Mathematik*, vol. 40, pp. 209–227, 1850.

[81] A. Seko, H. Hayashi, K. Nakayama, A. Takahashi, and I. Tanaka, "Representation of compounds for machine-learning prediction of physical properties", Nov. 26, 2016.

[82] Spherical harmonics - wikipedia, [Online]. Available at: `https://en.wikipedia.org/wiki/Spherical_harmonics` . Last accessed: Jun. 3, 2019.

[83] M. de Jong, W. Chen, R. Notestine, K. Persson, G. Ceder, A. Jain, M. Asta, and A. Gamst, "A statistical learning framework for materials science: Application to elastic moduli of k-nary inorganic polycrystalline compounds", *Scientific Reports*, vol. 6, p. 34 256, 2016.

[84] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", p. 26,

[85] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping", in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, 2001, pp. 402–408.

[86] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction", in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds., ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 758–770.

[87] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information", *Physical Review E*, vol. 69, no. 6, Jun. 23, 2004.

[88] P.-P. De Breuck. Supplementary website, [Online]. Available at: `pdebreuck.github.io` . Last accessed: Jun. 9, 2019.

[89] P.-P. De Breuck. Supplementary codes, [Online]. Available at: `https://github.com/pdebreuck/VibML` . Last accessed: Jun. 9, 2019.

[90] J. H. Friedman, "On bias, variance, 0/1—loss, and the curse-of-dimensionality", *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 55–77, Mar. 1, 1997.

[91] D. M. Hawkins, "The problem of overfitting", *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, Jan. 1, 2004.

[92] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overtting", p. 30,

[93] J. Reunanen, "Overtting in making comparisons between variable selection methods", p. 12,

[94] K. Kira and L. A. Rendell, "A practical approach to feature selection", in *Machine Learning Proceedings 1992*, Elsevier, 1992, pp. 249–256.

[95] P. Somol, J. Novovicova, and P. Pudil, "Efficient feature subset selection and subset size optimization", in *Pattern Recognition Recent Advances*, vol. 56, Feb. 1, 2010.

[96] F. õ. Chollet. (2015). Keras, [Online]. Available at: `https://keras.io`.

[97] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[98] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[99] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning", *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, May 2016.

[100] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?", in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 3320–3328.

[101] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada: IEEE, May 2013, pp. 7304–7308.

[102] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks", in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA: IEEE, Jun. 2014, pp. 1717–1724.

[103] M. Huh, P. Agrawal, and A. A. Efros, "What makes ImageNet good for transfer learning?", *arXiv:1608.08614 [cs]*, Aug. 30, 2016. arXiv: 1608.08614.

[104] Z. Li and D. Hoiem, "Learning without forgetting", *arXiv:1606.09282 [cs, stat]*, Jun. 29, 2016. arXiv: 1606.09282.

[105] F. Legrain, A. van Roekeghem, S. Curtarolo, J. Carrete, G. K. H. Madsen, and N. Mingo, "Vibrational properties of metastable polymorph structures by machine learning", *Journal of Chemical Information and Modeling*, vol. 58, no. 12, pp. 2460–2466, Dec. 24, 2018.

[106] P.-P. De Breuck. TTNN: A thermal transfer neural network, [Online]. Available at: https://github.com/pdebreuck/TTNN . Last accessed: Jun. 9, 2019.

[107] H. Wu, A. Lorenson, B. Anderson, L. Witteman, H. Wu, B. Meredig, and D. Morgan, "Robust FCC solute diffusion predictions from ab-initio machine learning methods", *Computational Materials Science*, vol. 134, pp. 160–165, Jun. 2017.

[108] Matminer (materials data mining) — matminer 0.5.4 documentation, [Online]. Available at: https://hackingmaterials.github.io/matminer/ . Last accessed: Mar. 5, 2019.

[109] D. W. Davies, "Materials discovery using chemical heuristics and high-throughput calculations", p. 205,

[110] P. Villars, K. Cenzual, J. Daams, Y. Chen, and S. Iwata, "Data-driven atomic environment prediction for binaries using the mendeleev number", *Journal of Alloys and Compounds*, vol. 367, no. 1, pp. 167–175, Mar. 2004.

[111] Cohesion in metals (1988 edition) - open library, [Online]. Available at: https://openlibrary.org/books/OL2529857M/Cohesion_in_metals . Last accessed: Mar. 7, 2019.

[112] H. Robbins and S. Monro, "A stochastic approximation method", *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[113] ADL. (Aug. 27, 2018). A brief introduction to reinforcement learning, freeCodeCamp.org, [Online]. Available at: https://medium.freecodecamp.org/a-brief-introduction-to-reinforcement-learning-7799af5840db . Last accessed: Apr. 1, 2019.

[114] E. L. Willighagen, R. Wehrens, P. Verwer, R. de Gelder, and L. M. C. Buydens, "Method for the computational comparison of crystal structures", *Acta Crystallographica Section B: Structural Science*, vol. 61, no. 1, pp. 29–36, Feb. 1, 2005.

[115] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, Feb. 1997.

[116] (Aug. 1, 2005). Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy, [Online]. Available at: https://www.computer.org/csdl/journal/tp/2005/08/i1226/13rRUy3xY96 . Last accessed: May 25, 2019.

[117] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data", in *Proceedings of the 24th international conference on Machine learning - ICML '07*, Corvalis, Oregon: ACM Press, 2007, pp. 759–766.

[118] E. Meighen and R. Yue, "Hybrids of chemical derivatives of escherichia coli alkaline phosphatase", *Biochimica Et Biophysica Acta*, vol. 412, no. 2, pp. 262–272, Dec. 15, 1975.

[119] OQMD, [Online]. Available at: `http://oqmd.org/` . Last accessed: Jun. 9, 2019.

[120] R. Kohavi and R. Quinlan, *Decision tree discovery*, Oct. 10, 1999.