Linköping University | The Department of Physics, Chemistry, and Biology
Master thesis, 30 ECTS | Applied Physics and Electrical Engineering - Theory, Modelling, Visualization
2017 | LIU-IFM/LITH-EX-A--17/3427--SE

Machine Learning of Crystal Formation Energies with Novel Structural Descriptors

Maskininlärning av kristallers formationsenergier

Claudia Bratu

Supervisor : Joel Davidsson
Examiner : Rickard Armiento

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Avdelning, institution
Division, Department

Theoretical Physics, Department of Physics, Chemistry and Biology, Linköping University

Datum
2017-11-23

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

ISBN

ISRN: LITH-IFM-A-EX--17/3427--SE

Serietitel och serienummer **ISSN**
Title of series, numbering _____

URL för elektronisk version
http://urn:nbn:se:liu:diva-143203

Titel
Machine Learning of Crystal Formation Energies with Novel Structural Descriptors

Författare
Claudia Bratu

Sammanfattning
To assist technology advancements, it is important to continue the search for new materials. The stability of a crystal structures is closely connected to its formation energy. By calculating the formation energies of theoretical crystal structures it is possible to find new stable materials. However, the number of possible structures are so many that traditional methods relying on quantum mechanics, such as Density Functional Theory (DFT), require too much computational time to be viable in such a project. A presented alternative to such calculations is machine learning. Machine learning is an umbrella term for algorithms that can use information gained from one set of data to predict properties of new, similar data. Feature vector representations (descriptors) are used to present data in an appropriate manner to the machine. Thus far, no combination of machine learning method and feature vector representation has been established as general and accurate enough to be of practical use for accelerating the phase diagram calculations necessary for predicting material stability. It is important that the method predicts all types of structures equally well, regardless of stability, composition, or geometrical structure. In this thesis, the performances of different feature vector representations were compared to each other. The machine learning method used was primarily Kernel Ridge Regression, implemented in Python. The training and validation were performed on two different datasets and subsets of these. The representation which consistently yielded the lowest cross-validated error was a representation using the Voronoi tessellation of the structure by Ward et. al. [Phys. Rev. B 96, 024104 (2017)]. Following up was an experimental representation called the SLATM representation presented by Huang and von Lilienfeld [arXiv:1707.04146], which is partially based on the Radial Distribution Function. The Voronoi representation achieved an MAE of 0.16 eV/atom at 3534 training set size for one of the sets, and 0.28 eV/atom at 10086 training set size for the other set. The effect of separating linear and non-linear energy contributions was evaluated using the sinusoidal and Coulomb representations. The result was that separating these improved the error for small training set sizes, but the effect diminishes as the training set size increases. The results from this thesis implicate that further work is still required for machine learning to be used effectively in the search for new materials.

Nyckelord
Machine learning, crystal, formation energy, kernel ridge regression, KRR, representation, descriptor, feature vector representation

Abstract

To assist technology advancements, it is important to continue the search for new materials. The stability of a crystal structures is closely connected to its formation energy. By calculating the formation energies of theoretical crystal structures it is possible to find new stable materials. However, the number of possible structures are so many that traditional methods relying on quantum mechanics, such as Density Functional Theory (DFT), require too much computational time to be viable in such a project. A presented alternative to such calculations is machine learning. Machine learning is an umbrella term for algorithms that can use information gained from one set of data to predict properties of new, similar data. Feature vector representations (descriptors) are used to present data in an appropriate manner to the machine. Thus far, no combination of machine learning method and feature vector representation has been established as general and accurate enough to be of practical use for accelerating the phase diagram calculations necessary for predicting material stability. It is important that the method predicts all types of structures equally well, regardless of stability, composition, or geometrical structure. In this thesis, the performances of different feature vector representations were compared to each other. The machine learning method used was primarily Kernel Ridge Regression, implemented in Python. The training and validation were performed on two different datasets and subsets of these. The representation which consistently yielded the lowest cross-validated error was a representation using the Voronoi tessellation of the structure by Ward et. al. [Phys. Rev. B 96, 024104 (2017)]. Following up was an experimental representation called the SLATM representation presented by Huang and von Lilienfeld [arXiv:1707.04146], which is partially based on the Radial Distribution Function. The Voronoi representation achieved an MAE of 0.16 eV/atom at 3534 training set size for one of the sets, and 0.28 eV/atom at 10086 training set size for the other set. The effect of separating linear and non-linear energy contributions was evaluated using the sinusoidal and Coulomb representations. The result was that separating these improved the error for small training set sizes, but the effect diminishes as the training set size increases. The results from this thesis implicate that further work is still required for machine learning to be used effectively in the search for new materials.

Acknowledgments

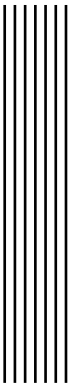
I would like to thank my supervisor Joel Davidsson and my examiner Rickard Armiento for their guidance and support during these past six months. Special thanks to Felix Faber, who went out of his way to help me with my endless stream of questions, even if it meant sacrificing his spare time. You three have together made this thesis possible.

Lastly, thanks to Isak Johansson-Åkhe, who has stood by my side and supported me through both good and bad times.

Contents

List of Figures

List of Tables



Preface

This thesis was done in the context of an ongoing research project, which has already produced one thesis [1], one smaller project report (of 15 ETSC) [2], and two journal papers [3], [4]. The long-term goal is to produce computationally efficient methods to use in the search for new materials. The short-term goal is to estimate the formation energy of different material combinations, that is, the energy of the structure compared to the 0 K structures of the included elements. Being able to estimate the formation energy is interesting because by studying the formation energy of different structures one might be able to predict what combinations the involved elements tend to form, and how the structures will be built up.

The formation energy can be quite accurately calculated through, e.g., Density Functional Theory (DFT), but methods based on quantum mechanics require a lot of calculation time. The time required can be more than what is viable when searching for new materials in a, typically, extensive space of chemical composition and structural configuration. In the earlier projects [1]–[4], it has been shown that machine learning (ML) is a possibly viable method of performing estimations of the formation energies of crystals.

1 Introduction

As science and technological advancements progress, there is a need to discover new materials with desired properties. New materials aid the creation of new technologies and improve already existing ones. These could be used in everything from building stronger buildings, creating water-resistant clothes, making solar cells, biodegradable plastics, and much more, including things we could not even imagine existing today. The materials available today restrict what we can do with our technology. Better materials will widen the possibilities and could help us improve living standards, save the environment, or aid space travel.

Finding these new materials is no simple task since there are so many different ways in which atoms bond and build up new structures. Also, most of the possible combinations will be unstable and decompose into other materials. Therefore, finding the stability of theoretical materials is important. The stability of a solid-state crystal structure is connected to its *formation energy*. The formation energy is the energy which the system gains by forming the new material as compared to the atoms being in their *elemental phases*. The elemental phase is the structure which an element forms at 0 K. At room temperature, atoms tend to seek arrangements that lowers the energy of the system. Structures that have a lower energy compared to other structures using the same elements are thus more likely to be stable. By calculating and comparing the formation energy of several crystal structures of a given elemental composition one can understand what structures that will possibly be stable.

There already exist good methods for calculating the formation energy of crystal structures. One such example is Density Functional Theory (DFT). However, while DFT is in general accurate enough to make predictive stability estimates through the formation energy, it is too slow for applications that need to screen through thousands of thousands of structures.

One approach to accelerate the calculation of formation energies is to use machine learning (ML). Machine learning is a collection of prediction methods that have been proved useful in a wide number of applications, ranging from computer vision to economics. The machine is given some data as a reference and proceeds to infer conclusions about the relation between different properties of the data. These relations can then be used to predict the properties of new data. In this thesis, machine learning will be used to predict the formation energies of solid-state crystals.

1.1 Results from the Earlier Projects

As mentioned in the preface, this thesis is one in a series of projects aiming to estimate the formation energy of crystals at 0 K. The ML method used in the earlier projects [1]–[4] was Kernel Ridge Regression (KRR) with a Laplacian kernel. This choice of method and kernel has been successfully used for calculating atomization energies of molecules [5], which makes it likely to be a good fit for calculating formation energies of crystals since the purpose is quite similar. The way the data is presented to the machine is also extremely important, as this affects the ability of the machine to create mathematical relations between the input and the formation energy. Some different data representations have been investigated, and so far, within the present project for applications involving different crystal structures, a sinusoidal representation has worked best. Chapter 3.1 covers more on the representations.

The machine predictions using the sinusoidal representation have been compared to values calculated previously with DFT, and the results have been promising. However, the error (mean absolute error, MAE) of the energies is still too large for the method to be really useful. One possible reason for the reported MAE being larger than in previous works on, e.g., molecules is because of the composition of the dataset used to train and evaluate the machine. The dataset originally used was the MP-basic, defined in Ref. [1], which was constructed using the Materials Project (MP) database [6]. This set contained a large number of elements, and a large number of types of crystal structures, creating a dataset seemingly more diverse than the set of small molecules with a few participating species used in prior studies. Hence, one theory was that fewer amounts of elements or fewer types of crystal structures would lower the error, as the method can then train on more similar structures. This would create a method which is more specialized on a certain type of structures, but which hopefully also would be more accurate. Because of this, only a specific subgroup of MP-basic, called MP-139, was tested separately. This did improve the error, but MP-139 is a quite small dataset (1312 crystals), which limits how well trained the machine can become.

The goal is to get the error down to 0.1 eV/atom. This value is chosen based on two things. First, this value is roughly the error of performing crude DFT calculations. Secondly, if the error is larger than 0.1 eV/atom then the noise of the formation energies will be too large, and it will be difficult to determine which structures that are likely to be stable or unstable. The best result from the first project [1] was obtained using a training set size of 500 crystals from the MP-139 dataset, and the error received was 0.25 eV/atom. This result was obtained using an additional ML step containing a preparatory machine learning (PML), which separates the formation energy into linear and non-linear parts. It has been shown that using the PML lowers the error. For example, the error for MP-basic at 500 training set size was around 1.5 eV/atom without the PML but below 1 eV/atom with PML. Similar behaviour was shown for other datasets.

The follow-up projects [2]–[4] experimented with performing ML on different types of data. Note that none of these projects used PML. First off was what here will be called the FLLA set, which consists of data collected from the Material Projects database [6]. FLLA obtained an error of 0.37 eV/atom at a training set size of 3000 [3], which came to be used as a reference point for later projects.

One of the projects [4] used a dataset containing only a single crystal structure, but many different elements. This achieved an error of 0.1 eV/atom for a training size of 10 000 crystals. This result was satisfying but of limited use as investigating formation energies of all possible substitutions into one and the same structure is not the most usual application.

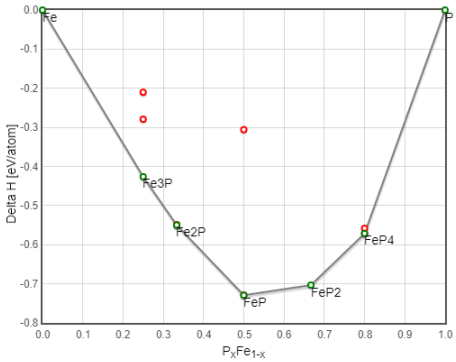
The latest project [2] did the opposite, using a dataset containing few elements, but many different structures. Many of these structures were purely theoretical and most likely unstable. The expected result was that the error would, at least, drop lower than the previous result of 0.37 eV/atom using FLLA. However, this was not the result. Instead, the opposite happened: the error became larger. Suggested reasons for this result were that the new dataset was still not restricted enough (and unrestricted in a different way from FLLA), or

that the chosen data representation (the sinusoidal representation) contained too little information, which would cause poor predictions.

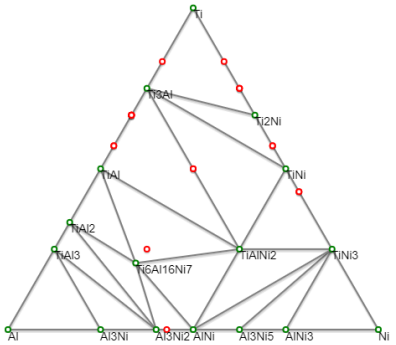
1.2 Phase Diagrams

Studying the formation energy of structures containing few elements is especially interesting as the results can be used to create *phase diagrams*. There are many types of phase diagrams, but the ones referred to in this report are shown in figure 1.1 and describe the stability of structures using different combinations of elements. The corners of the diagram represent structures containing only a single element. The length or area in between these represent all combinations of the elements, with each point representing a new structure. The larger the amount of a specific element in a structure, the closer its point will be to the corresponding corner. The formation energy of the structure is plotted in an additional dimension. In the case of a phase diagram using only two elements, see figure 1.1 a), the energy becomes the second dimension. In a phase diagram with three components, see figure 1.1 b), the energy becomes the third dimension. This extra dimension separates structures with identical composition but different structures as these could have different formation energies.

Phase diagrams of this kind are used to show what materials that are possible to form from a specific set of elements. Having theoretical phase diagrams may aid in the search for new materials by providing a hint of what materials may or may not exist. If the phase diagrams are accurate, then less time will be spent researching dead ends. Through machine learning, it may be possible to greatly reduce the amount of computational time needed to create such diagrams. The catch is that the machine must be good at predicting the energy of both thermodynamically stable (real) and unstable (theoretical) structures.



(a) A phase diagram using two elements (Fe and P), with the formation energy as the second dimension.



(b) A phase diagram using three elements (Ti, Al, and Na). Here the energy would be portrayed in a third dimension.

Figure 1.1: Examples of phase diagrams. The green dots represent stable (low-energy) structures while the red ones are unstable. Unstable compounds either has a stable (green) phase at the exact same composition but with lower formation energy, or can be phase-separated into a linear combination of stable phases at other compositions that give a lower total formation energy. The diagrams were generated with the Open Quantum Materials Database (OQMD) [7], [8].

1.3 Where Do We Stand?

The current goal is to find a machine learning method and representation that predicts with an error lower than 0.1 eV/atom consistently, regardless of data used (stable, unstable, any

sizes or compositions). Several sources (for example: Ref. [4], [9] and [10]) have reported errors around or smaller than 0.1 eV/atom in specific cases, but there has been no method that has been shown to perform that well consistently.

1.4 Aim

This project aims to investigate how accurate and useful different ML models are for accelerating the creation of phase diagrams in order to estimate the stability of crystal structures. First of all, the results from the latest project [2] will be reproduced. After that, the performance of different representations applied to the FLLA set and to the dataset from the latest project (the TAATA set) will be compared with prior work and each other. The datasets are described in section 4. The investigated representations are the sinusoidal representation [1], [3] (with small variations), some simple composition-based models [9], the Voronoi representation [10], the PRDF representation [10], [11], and the SLATM representation [12]. These will be detailed in chapter 3.1. Lastly, PML will be applied to the sinusoidal representation, and its effect on the ML performance will be evaluated.

1.5 Research Questions

1. Does the error indeed become larger for the TAATA set than for the FLLA set using the sinusoidal representation, and if yes, why?
2. How do the different representations perform when applied to the FLLA and TAATA sets?
    - How well do they compare to each other overall?
    - Is there an individual difference in performance between the FLLA set and the TAATA set? If yes, how come?
3. Does PML improve the performance of the sinusoidal representation?
4. Does any of the tested representations achieve the goal of an MAE lower than 0.1 eV/atom with a reasonable amount of training data?

1.6 Delimitations

It is easier to compare the representations with each other if the ML method stays constant. Therefore, the primary ML method of this project will be Kernel Ridge Regression (KRR), and the kernel used will be the Laplacian kernel. These have continued to performed well not only for molecules but also for crystals, as seen in the earlier projects. Other methods will only be used in very specific cases.

The ML method will be used on some specific datasets: the FLLA set, and the TAATA set (containing three subsets, TAATA-Hf, TAATA-Ti, and TAATA-Zr). It is common that a ML method or representation works well for one dataset, but not for another. Therefore, it is important to distinguish which dataset was used for training and validation.

1.7 Definitions

CV = cross-validation

KRR = Kernel Ridge Regression

MAE = Mean Absolute Error

ML = Machine Learning

ML method = the method of applying machine learning. Examples include KRR, RF. Sometimes this also refers to a ML method combined with a specific data representation.

MP = Materials Project

PCA = Principal Component Analysis

PML = Preparatory Machine Learning

PRDF = Partial Radial Distribution Function

RF = Random Forest

SLATM = Spectrum of London and Axilrod-Teller-Muto potentials

Also, if nothing else is stated, "error" refers to the MAE, and is given in eV/atom.

2 Machine Learning

This chapter aims to explain how machine learning works and what methods are used in this thesis. Some ways to improve the machine learning performance, and some traps to watch out for, are also included. The subsequent chapter will closely follow the formalism and notation used in Ref. [1] and Ref. [3].

2.1 Kernel Ridge Regression

The basic idea of machine learning is that there are two sets of data, $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ and $\boldsymbol{y} = (y_1, y_2, ..., y_n)^T$, which correspond to input and output. The issue at hand is to find the function f which approximates the output $\boldsymbol{y}$ from the input $\boldsymbol{x}$, that is:

$$f(\boldsymbol{x}) = \boldsymbol{y}. \tag{2.1}$$

This is done with the help of computers via a machine learning algorithm (also called machine learning method). There are many different methods, and one of these is the *Kernel Ridge Regression*, shortened KRR. This method is an extension of the Ridge Regression method.

Ridge Regression

Ridge Regression solves this problem by searching for a function on the form

$$\boldsymbol{y} = f(\boldsymbol{X}, \boldsymbol{w}) = \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w} = \sum_j h(\boldsymbol{x})_j w_j, \tag{2.2}$$

where $\boldsymbol{w} = (w_1, w_2, ..., w_q)^T$, $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)^T$ with each $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})$, meaning that $\boldsymbol{X}$ is a nxp-matrix. n is the number of data points, and p is the number of properties identifying each data point. $h(\boldsymbol{x})$ is an operator which acts on a vector $\boldsymbol{x}$ and returns a new vector with terms according to some chosen function, usually a polynomial.

For example, if $h(\boldsymbol{x})$ is chosen to create terms according to a second-degree polynomial and acts on the vector $\boldsymbol{x} = (A, B, C)$ then $h(\boldsymbol{x}) = (1, A, B, C, AB, AC, BC, A^2, B^2, C^2)$. $\boldsymbol{w}$ becomes the coefficients for the polynomial:

$$y = \sum_j h(x_j)w_j = w_1 + w_2 A + w_3 B + w_4 C + w_5 AB + ... + w_{10} C^2. \tag{2.3}$$

$h(\boldsymbol{X})$ is then a nxq-matrix where q is the length of $h(\boldsymbol{x})$, which depends on the chosen function and the number of properties p. Since the operator $h(\boldsymbol{x})$ is chosen, the problem is then simply to find $\boldsymbol{w}$.

The normal equation can give an approximative solution to this type of problem, which in this case leads to the solution

$$\boldsymbol{w} = (\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{h}(\boldsymbol{X}))^{-1} \boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{y}. \tag{2.4}$$

The normal equation minimizes the Euclidian norm $||\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}||_2$. Since $\boldsymbol{w}$ now is known, and $\boldsymbol{h}(\boldsymbol{X})$ is chosen, we have all we need to calculate $\boldsymbol{y}$ from a given $\boldsymbol{X}$. This can now be used to predict new points of data, based on the old points. The data points used to build a model (by calculating $\boldsymbol{w}$) are called the *training set*, while data points used to estimate the performance of the model are called the *test set* or *validation set*.

To avoid *overfitting*, the error $||\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}||_2$ is penalized by adding an additional term. The error to minimize then becomes $||\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}||_2^2 + \lambda||\boldsymbol{w}||_2^2$, where λ is a constant. The norm is squared because this makes the calculation easier. The end result is the same as the solution that minimizes the norm also minimizes the squared norm. Overfitting means that the function $f(\boldsymbol{X})$ is more complex than what is necessary, see figure 2.1 b). Its opposite is called *underfitting*. Overfitting will cause the function to fit very well to given data points, but to fit very poorly to new, similar data points. Underfitting will essentially do the opposite, moving towards an average of the given data points, see figure 2.1 a).

To find the $\boldsymbol{w}$ that minimizes the error, we take the gradient of the error function with respect to the variables $\boldsymbol{w}$ [13]:

$$\nabla(||\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}||_2^2 + \lambda||\boldsymbol{w}||_2^2) = \vec{0}, \tag{2.5}$$

where

$$\begin{aligned}
&\nabla(||\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}||_2^2 + \lambda||\boldsymbol{w}||_2^2) = \\
&\nabla((\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}) \bullet (\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}) + \lambda \boldsymbol{w} \bullet \boldsymbol{w}) = \\
&2 \cdot \nabla(\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}) \bullet (\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}) + 2\lambda \boldsymbol{w} = \\
&2 \cdot -\boldsymbol{h}(\boldsymbol{X})^T \bullet (\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w}) + 2\lambda \boldsymbol{w} = \\
&-2\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{y} + 2\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w} + 2\lambda \boldsymbol{w}.
\end{aligned} \tag{2.6}$$

Hence,

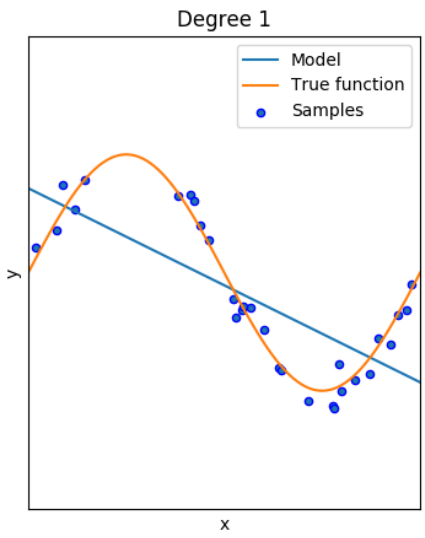
$$\begin{aligned}
-\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{y} + \boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w} + \lambda \boldsymbol{w} = \vec{0} \quad &\Rightarrow \\
-\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{y} + (\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{h}(\boldsymbol{X}) + \lambda I)\boldsymbol{w} = \vec{0} \quad &\Rightarrow \\
\boldsymbol{w} = (\boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{h}(\boldsymbol{X}) + \lambda I)^{-1} \boldsymbol{h}(\boldsymbol{X})^T \boldsymbol{y}.&
\end{aligned} \tag{2.7}$$

The Kernel Trick

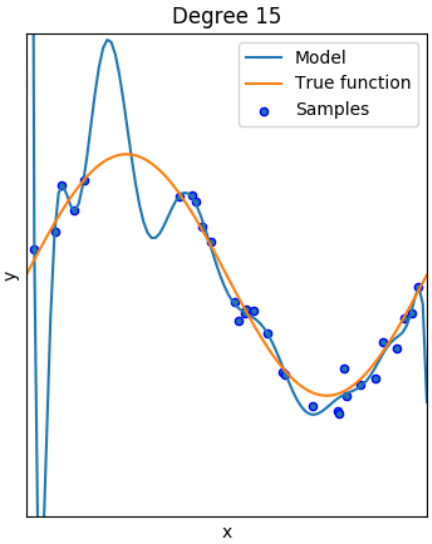
More complicated problems are usually non-linear. This makes it difficult to choose a proper $h(\boldsymbol{x})$ for such a problem. To make the problem simpler, one can map the data points $\boldsymbol{x}_i$ from the non-linear space to a linear space. This would make it much easier for the ML algorithm to accurately predict the output. However, in the same way that it is a problem to choose a good function $h(\boldsymbol{x})$, it will now be a problem to choose a proper mapping.

Thankfully, the ML algorithm only relies on the dot product between the data points $\boldsymbol{x}$, which in a mapped problem would be the dot product of the mapped data points $\Phi(\boldsymbol{x})$, if Φ is our map. Therefore, we need not know the map directly, it is enough to know the dot product [15, pp. 436-437]. This is where the so-called *kernel trick* makes its entrance [15, pp. 436-437], [16]. It can be shown that a type of functions called *kernels* can replace the dot product:

$$\langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.8}$$



(a) Underfitting. The order of the polynomial is too low to match the given data.



(b) Overfitting. The order of the polynomial is unnecessarily high, making it bad at predicting new data points.

Figure 2.1: Two different polynomial functions attempting to fit a number of data points of $sin(2x)$ with some added noise. Generated using code from Ref. [14]

Here, $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernel. In some cases, it can also be more computationally efficient to calculate the kernel function instead of performing the dot product.

A kernel is also a type of *similarity measure* and tells us how "similar" two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are [16]. There are many different kinds of kernels. In this thesis, the *Laplacian kernel* will be used. It is defined as

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{-||\boldsymbol{x}_i - \boldsymbol{x}_j||_1}{\sigma}\right), \tag{2.9}$$

where $||\boldsymbol{x}_i - \boldsymbol{x}_j||_1 = \sum_k |\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}|$ is the Manhattan norm. The hyperparameter σ is called the *kernel width*.

Ridge Regression of the mapped problem then becomes

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \sum_j k(\boldsymbol{x}, \boldsymbol{x}_j)\alpha_j \tag{2.10}$$

with

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda I)^{-1}\boldsymbol{y}, \tag{2.11}$$

where $\boldsymbol{K}$ is the *kernel matrix* (or the *Gram matrix*), $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\boldsymbol{\alpha}$ is a vector with a similar purpose to $\boldsymbol{w}$ in Ridge Regression, but with different dimensions. In our case, $K_{ij} = K_{ji}$, so $\boldsymbol{K}$ will be a symmetrical matrix.

This method, Ridge Regression with the help of kernels, is what is called *Kernel Ridge Regression*.

2.2 Random Forest

Another ML algorithm is the *Random Forest* (RF) [10], [17]. Random Forest solves the problem of finding f in a very different manner as compared to KRR. The algorithm creates a decision

tree with different end values at the leaves. When a data is to be predicted it is run through the tree, and the predicted answer is the value at the leaf which the algorithm ends up at. Essentially, this means that the algorithm groups the data based on their properties (values of $\boldsymbol{x}_i$) and the target property (y), putting data with similar values of the target property in the same group.

Several decision trees are created from random subsets of the data, creating a "forest" of decision trees. The data to be predicted is run through all these trees and all answers are weighed together to create a final prediction.

2.3 Cross-validation

In machine learning, so-called *hyperparameters* such as the overfitting constant λ, or the kernel width σ, must be chosen more or less by trial and error, helped by intuition and experience. However, searching for an optimal value of a hyperparameter (the value that yields the smallest error) while using only one training set and one test set will result in a hyperparameter value that works perfectly fine for *these* specific two sets, but that may give poor results if applied to anything else.

To avoid this, a scheme called cross-validation (CV) is introduced. Cross-validation uses a single dataset to create different combinations of training and test sets that all will help decide the best value of the hyperparameter. There are different types of cross-validation. The ones relevant to this work is *k-fold cross-validation* and *Monte Carlo cross-validation*.

k-fold Cross-validation

k-fold CV uses the following scheme [5]:

1. Randomly split the data into k equally large datasets.
2. Use one dataset as the validation set, and the rest form the training set.
3. Repeat this procedure k times, until each subset has acted validation set once.
4. Combine the k estimates of the prediction error into a single *cross-validation error*, for example through the mean.

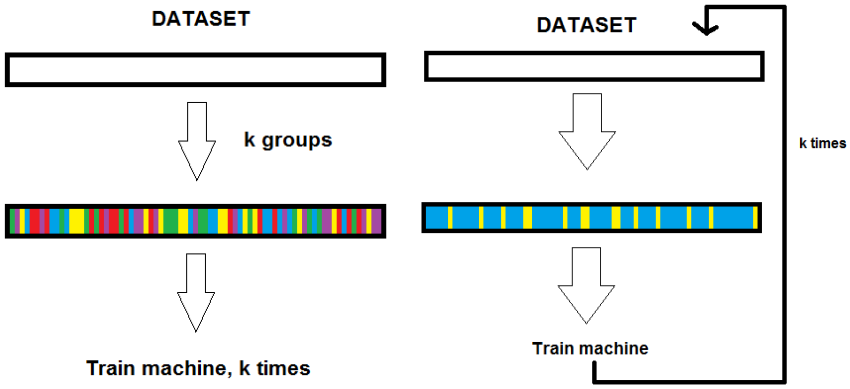
This scheme may also be repeated several times, randomly splitting the dataset again and again. Figure 2.2 a) shows how the data is split into different groups and then trained. The drawback of cross-validation is that using it takes a significant amount of time. Instead of building a single ML model, k ones are built.

Performing a k-fold cross-validation ensures that the hyperparameter value is decent for not only a single test set, but for most test sets that could be built from the original dataset. However, the goal of creating ML models is to be able to predict new data fairly well. If the chosen model (which is defined by chosen hyperparameter values and ML method) is built to work well for a dataset then, of course, all tests using that dataset will yield good results. This is essentially the same effect as picking optimal hyperparameter values for the dataset based on specific training and test set, but this time the effect is more subtle, and not as extreme.

Therefore it is recommended to add an additional layer of cross-validation. Before the scheme described above is applied, a part of the dataset is set aside as the final test set. Then the cross-validation scheme is applied to find the best model (hyperparameter values) for the rest of the dataset, and the final result is given by applying the model to the final test set.

Setting aside part of the dataset before performing the cross-validation simulates using the built model on completely new data where the sought property is unknown. We validate the model using part of our dataset only in order to get an estimate of how well our model would perform in an actual application. This method may be repeated so that different parts of the

dataset get to act as the final test set, just like the cross-validation for finding hyperparameter values.



(a) *k*-fold cross-validation. (b) Monte Carlo cross-validation.

Figure 2.2: The difference between the two types of cross-validation. The colours represent groupings of data.

Monte Carlo Cross-validation

Monte Carlo cross-validation, or *repeated random subsampling* [18], is similar to *k*-fold CV but differs on one point: the size of training and test size is not determined by the number of folds. Instead, a fixed training set size n is chosen, and that many data are randomly selected from the full dataset. All other data will be used as test set. This scheme is also repeated k number of times, each time picking n random data for the training set. Figure 2.2 b) shows how the data is split and trained over and over again. This method has the effect that, in contrast to *k*-fold cross-validation, some data points may be used several times as test data, and some might never be used at all.

2.4 A Word on Machine Learning Bias

Probably the largest problem with machine learning is bias towards specific types of data. How well the machine predicts is first and foremost dependent on the training data used. ML method and data representation come second. For example, if a machine learning algorithm is trained to recognize pictures of cats, but almost all pictures presented in the training set are of long-haired cats, then the machine will most likely not recognize short-haired cats as cats. Even worse, it might recognize long-haired dogs or guinea pigs as cats if it connects the property "cat" with the property "long hair". The same principle applies to all types of machine learning. The training set must be representative for what the machine is supposed to predict.

In this thesis, the focus lies on applying machine learning to crystals. Similar issues as those described above could be having a bias toward specific elements, crystal structures, unit cell sizes, or formation energies. Because of this, one must always be aware of the possible shortcomings of the final model due to the type of data used.

2.5 Implementation of the ML Method

As explained in chapter 1, this project has two main goals: to reproduce earlier results and to compare the sinusoidal representation with other competing representations using a selection of datasets. The ML method primarily used was KRR (described in chapter 2.1), implemented

in Python. The final validation was done using Monte Carlo cross-validation, as this method was most similar to the cross-validation used in the earlier projects [1]–[3].

The KRR was implemented twice, first from scratch, and later using functions from the library QML (Quantum Machine Learning) [19]. Both implementations gave the same result. The results from the implementation were continuously compared to earlier results from Ref. [1] and Ref. [3] to ensure that the method was working as expected.

All representations except two (Voronoi and PRDF, see chapter 3.1 or 3.2) used this ML model. The other two used separate Random Forest and Kernel Ridge Regression implementations from the supplement code to Ref. [10]. The reason for this is that these two representations were already implemented using these ML methods and another programming language. This made it easier to use these ML methods directly instead of transferring the representations to Python and applying the above KRR to them. These implementations utilized 10-fold CV instead of Monte Carlo CV and the PRDF representation used a kernel with a Frobenius norm instead of a L1-norm.

The KRR from Ref. [10] was compared to the one created in this project using the FLLA set, and the two gave very similar results. The small differences that were present are considered small enough to not interfere with any conclusions drawn within this project. These are assumed to stem from small implementation differences, such as functions, data types, and programming language used.

Training on Deviation

As stated in chapter 2.1, the kernel function in KRR acts as a sort of similarity measure, and tells us if two vectors are "similar" or not, and how similar they are. The kernel will return a value between 0 and 1. The smaller the difference between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (that is, the smaller $||\boldsymbol{x}_i - \boldsymbol{x}_j||$ is), the closer to 1 the result is. This value is modified by the kernel width, σ. The smaller σ is, the faster the kernel value drops towards zero as the difference $||\boldsymbol{x}_i - \boldsymbol{x}_j||$ increases. This is why the kernel width is very important, it regulates how easily the function considers two vectors as similar.

Since the predicted value depends heavily on the kernel function (see equation 2.10), if there is a too large difference between the data to be predicted and the data used as training set, the result will simply become zero. This requires that the prediction data point fails to be picked up by any data point in the whole training set.

A prediction of zero is likely to be very far from the true value, and therefore a very bad prediction. An option is to perform machine learning not on the formation energy straight off, but instead train it on the deviation from the mean energy of the training set. The mean is then added onto the predicted deviation. In practice, this would cause the method to predict outliers to have a formation energy equal to the mean energy of the training set, instead of zero. This method has been applied to almost all KRR results in this report. The exceptions are the PRDF representation and the Voronoi representation since these were using a separate implementation. The tests using preparatory machine learning (see below) does not use deviation training, as there is a risk that this causes problems for the preparatory ML.

Preparatory Machine Learning

Another possible method of improving the result of the KRR prediction is assuming that there are two types of contributions to the formation energy of a crystal [1]. The first type is assumed to be a linear energy contribution from each atom in the unit cell, and the second is assumed to be a non-linear energy contribution resulting from the pair interaction between all atoms. That is: $y_{tot} = y_{linear} + y_{non\text{-}linear}$. The machine learning is mostly needed to deal with the non-linear terms, so a preliminary step that separates the linear and non-linear terms may be introduced.

Ridge Regression is used to fit a linear function to the energy using only the constituent atoms as input:

$$f(\boldsymbol{X}, \boldsymbol{w}) = \boldsymbol{h}(\boldsymbol{X})\boldsymbol{w} = \boldsymbol{X}\boldsymbol{w} = \sum_j \boldsymbol{x}_j w_j, \tag{2.12}$$

when $h(\boldsymbol{x})$ is chosen as an operator which returns terms for a linear polynomial.

Here $\boldsymbol{x}_j$ is something representing the atoms included in the structures. In this project, as in the previous [1], this representation was a vector where the value at index i explains how many atoms with atom number i that were present within the structure. Index value 0 was always set to 1. For instance, if a structure would contain three hydrogen atoms, four beryllium atoms, two carbon atoms, and one oxygen atom, then the corresponding representation would be the vector

$$\boldsymbol{x}_{struct} = (1, 3, 0, 0, 4, 0, 2, 0, 1, 0, 0, 0, ...). \tag{2.13}$$

$f(\boldsymbol{x}_{struct})$ would then be calculated through

$$f(\boldsymbol{x}_{struct}) = w_0 \cdot 1 + w_1 \cdot 3 + w_4 \cdot 4 + w_6 \cdot 2 + w_8 \cdot 1. \tag{2.14}$$

By using machine learning, an $\boldsymbol{w}$ that gives a good approximation for the training data can be calculated. This linear method of approximating the formation energy will hopefully catch the linear contributions from the different atoms, allowing us to define the linear contribution as $y_{linear} = \boldsymbol{X}\boldsymbol{w}$.

The linear contribution can be ignored for the time being, and the KRR is only passed $y_{non\text{-}linear}$. There are then two steps to consider when making predictions: one for the linear contribution, and one for the non-linear contribution. This method has shown some potential for improving the accuracy of KRR predictions. In this report, it will simply be called "Preparatory machine learning" (PML).

A small change that can be done to the PML would be to switch the first element from 1 to 0, ensuring that all contributions to the linear energy come from the atoms, with no additional constants. For simplicity, this alternative will be referred to as "Alternative PML".

3 Data Handling

This chapter introduces the representations used in the thesis, what was compared, and how the representations were implemented.

3.1 Representations

What data is given to the ML method, and how this data is presented, may affect how well the machine learns and predicts. Without relevant data, it cannot draw good conclusions about the relations between input and output. The same holds if the data is represented in a way not fitting for the algorithm, even if the data is relevant. The given data, and the way it is presented to the machine is called the *descriptor* or the *feature vector representation*. In this thesis, this will be shortened to *representation*. This section will focus on the different representations that have been used in this thesis. Most of these representations are specifically designed to describe crystal structures.

Coulomb Representation

The *Coulomb representation* is a simple and well-used representation for molecules [5], [20], [21]. A matrix is built up as:

$$C_{ij} = \begin{cases} \frac{Z_i^{2.4}}{2}, & \text{if } i = j \\ \frac{Z_i Z_j}{||\boldsymbol{r}_i - \boldsymbol{r}_j||_2}, & \text{if } i \neq j \end{cases}, \tag{3.1}$$

where Z_i is the atomic number of atom i and $\boldsymbol{r}_i$ is a vector describing the position of atom i. The diagonal element $\frac{Z_i^{2.4}}{2}$ is an approximation of the total energy of the free atom i, and the non-diagonal element $\frac{Z_i Z_j}{||r_i - r_j||_2}$ is the Coulomb repulsion between atom i and j. To make the Coulomb representation equally large for all sizes of molecules the matrix is also padded with zeroes up to the maximal size for the molecule present in the current dataset. This corresponds to filling the molecule with atoms that have no charges, leading to no energy contribution and no interaction between any of these atoms with any other. To make the Coulomb representation invariant of different ordering of the atoms, the rows and columns can be sorted after their norm.

Sinusoidal Representation

The *Sinusoidal representation* [1], [3] is based on the Coulomb representation but is modified to be used with crystals instead of molecules. Some terms are modified to account for the periodicity of the crystals as compared to molecules. Like the Coulomb representation, the sinusoidal representation should be sorted. The matrix elements are modified to:

$$C_{ij} = \begin{cases} \frac{Z_i^{2.4}}{2}, & \text{if } i = j \\ \frac{Z_i Z_j}{||\boldsymbol{M}\sin^2(\pi(\boldsymbol{q}_i - \boldsymbol{q}_j))||_2}, & \text{if } i \neq j \end{cases}, \tag{3.2}$$

where Z_i is the atomic number of atom i, $\boldsymbol{M}$ is the basis matrix of the lattice, and $\boldsymbol{q}_i$ is the position vector of atom i in the basis of the crystal lattice.

Representation by Composition Only

The sinusoidal representation is an example of a representation that takes both the composition and the structure of the crystal into account. Recent studies have experimented with providing only the composition of a crystal, and have gotten fairly good results. Linear models developed by Deml et. al. [9] have achieved low MAE values, and are said to be able to accurately predict both elements not included in training and experimentally unreported compounds.

The models were developed through a systematic method of including and removing properties used as input to a machine learning algorithm. The most important properties for predicting the formation energy of different compounds could then be determined to form a final model. Some of the more important properties were the total energies of the constituent elements in their elemental phases, atomic ionization energies and electron affinities, Pauling electronegativity differences, and atomic electric polarizabilities.

The results were two different linear models which can be found in the supplementary material to Ref. [9]. These models were constructed for a dataset of compositionally diverse, stable, metal-nonmetal compounds and a subset of chalcogenides, respectively.

Voronoi Representation

Another approach by Ward et. al. [10] adds Voronoi tessellation as a central element to the representation of crystals. In total, this representation takes 271 different properties into account for each crystal. The representation has been used to show that both composition-based and structure-based attributes are important to achieve as low error as possible [10].

In the original work, this representation has been tested both with KRR and Random Forest, and so far has shown best results with the Random Forest. It has been compared to the sinusoidal representation and shows promising results so far (see figure 3 in Ref. [10] [1]).

PRDF

The *Partial Radial Distribution Function (PRDF) representation* [11], uses a version of the Radial Distribution Functions (RDF) as a means to describe crystal structures. The representation calculates the density of atoms of type β at a distance r in a band of width dr centred around an atom of type α, e.g. the density of β-atoms in a shell centred around the chosen atom. This density is then averaged over all atom types. For a specific r, the representation is given by:

$$g_{\alpha\beta}(r) = \frac{1}{N_\alpha V_r} \sum_{i=1}^{N_\alpha} \sum_{j=1}^{N_\beta} \theta(d_{\alpha_i\beta_j} - r)\theta(r + dr - d_{\alpha_i\beta_j} - r), \tag{3.3}$$

[1] In this work the sinusoidal representation is called "Coulomb Matrix", or "CM".

where N_α and N_β are the number of atoms of type α and β, respectively. V_r is the volume of the shell at distance r with thickness dr (the volume of the area under consideration). $\theta(x)$ is the *Heaviside step function*. The full representation is given by a matrix with elements $x_{\alpha\beta,n} = g_{\alpha\beta}(r_n)$, where the radius r_n is varied discretely between 0 and up to some cut-off radius r_{cut}, which is given as a hyperparameter.

SLATM

The *SLATM (Spectrum of London and Axilrod-Teller-Muto potentials) representation* [12] consists of three parts: information about single atoms, atom pairs, and atom triplets. Single atoms are simply described by their atomic number. To describe pairs of atoms and their interaction, SLATM employs the RDF in a similar manner to the PRDF representation. RDF describes how the atom density varies as a function of the distance from a reference particle, telling us something about the structure of the crystal. The three-body part is added to this to bring extra information about the structure (among other things, by describing angles within the structure). This part helps, for instance, to differ between homometric compounds.

3.2 Representation Variants

In this work, a number of the above representations, some with small variations, are tested and compared to each other. The Voronoi, PRDF, and SLATM representations are used directly as-is, with no variations.

Sinusoidal Representation

The basic sinusoidal representation is described in chapter 3.1. In this thesis, some variants of the sinusoidal representation are proposed and investigated. These are in this work referred as *complemented* variants.

The complemented variants add different information to the representation, such as the periodic table row and column number of the atoms in the structure. "Atoms per volume" refers to the number of atoms of atom type i divided by the total volume of the unit cell (that is, a vector with an element for each atom type). "Normalized atoms per volume" is the exact same thing, but the vector has then also been normalized. Note that in the normalized case, the dependence on the volume effectively disappears. By adding extra properties to the representation, it should be possible to determine if they have a significant impact on the formation energy.

In order to determine if either version of PML has an impact on the prediction accuracy, these are applied to the sinusoidal representation.

Composition Only

As using the scheme from Ref. [9] (see chapter 3.1) to create new models for our datasets would take far too much time, two alternative approaches are used. First, the pre-made models taken directly from the supplement of Ref. [9] are applied to our datasets. These two models are named model 1 and model 2 in the supplement, and these names are also used in this report. After that, the same input variables that are used in these models are used as input to a KRR using our datasets, to see if these inputs are generally important for predicting the formation energy. As model 1 and model 2 use different crystal properties (a result of the scheme applied to create them, see chapter 3.1), these two are tested separately here as well.

3.3 Summary of Investigated Represention

Below is a list of all representations investigated in this work, including their variants.

- Sinusoidal representation
  - Basic
  - Complemented with information about:
    - Row and column of atoms
    - Row, column, atoms per volume
    - Row, column, normalized atoms per volume
    - Row, column, atoms per volume, normalized atoms per volume
  - With PML
    - Normal PML
    - Alternative PML
- Composition only
  - Pre-made models
    - Model 1
    - Model 2
  - With KRR
    - Model 1
    - Model 2
- Voronoi representation
- PRDF
- SLATM

3.4 Implementation of the Representations

The sinusoidal representation was fully implemented in Python as part of this work with and was sorted using the L1-norm (Manhattan norm). The composition-only representation was implemented via the Python code found in the supplement of Ref. [9]. The Voronoi and PRDF representations were implemented via the Bash/Magpie/Java code in the supplement to Ref. [10]. Lastly, the SLATM representation was implemented using the Python library QML [19]. The raw data was given to the program in the POSCAR file format.

3.5 PCA

Principal Component Analysis, shortened to *PCA*, is a method of analyzing data. It is especially useful for analyzing data of high dimension, as it is possible to reduce the dimensionality of the data to the desired amount, while still keeping as much information as possible.

PCA calculates the variance of different variables in data inputs, or linear combinations of those variables, and seeks to find the set of new variables (linear combinations) that give the highest variance while still keeping all the new variables uncorrelated to each other.

In this thesis, we perform a 2D PCA, which means that we select only the two components that give the largest variance. By doing so, it is possible to plot the two components against each other, giving us a visual representation of the variance in a set of data. The axes of a PCA plot will be treated as dimensionless, as the data has been transformed beyond recognition. The numbers themselves tell very little, but the spread of the data points makes it possible to understand how similar data points are to each other, and more so, makes it easier to spot data points that are very different from the rest.

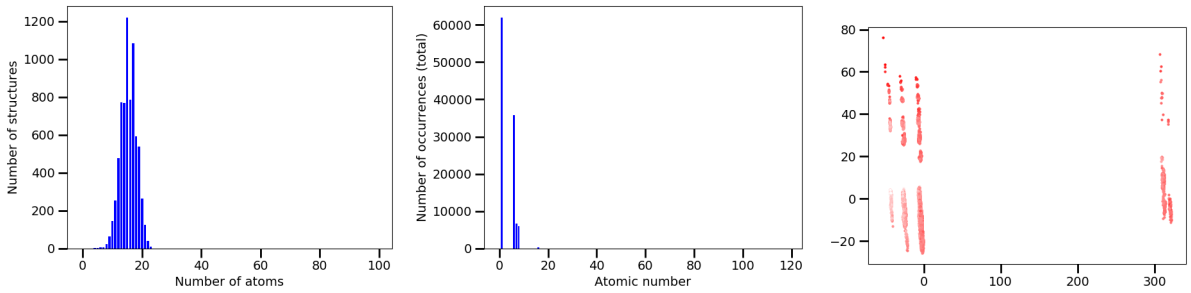
A more detailed explanation of PCA can be found, for example, in Ref. [22].

4 Datasets

This section describes the different datasets used in the project.

4.1 QM7

QM7 is a subset of GDB-13 (a database of organic molecules), which contains molecules of up to 23 atoms [23]–[25]. QM7 contains 7165 molecules, composed of elements H, O, S, N, and C. The molecules in the dataset are relaxed to a local minimum energy state [1]. The lowest formation energy found in the dataset is -95.12. This dataset has previously been used to compare machine learning methods and is therefore a good dataset to use for comparison [5]. The size distribution, element distribution, and 2D PCA of QM7 is shown in figure 4.1. The colour intensity of the dots in the PCA plot matches the energy of the plotted structure. Darker dots represent a high (possibly positive) formation energy while lighter dots represent a low energy. The element distribution counts the total amount of occurrences. This means, for example, that if oxygen would be present five times within a single structure, then the contribution to the occurrence would be five, and not one. The QM7 set is only used when making comparisons with previous results.

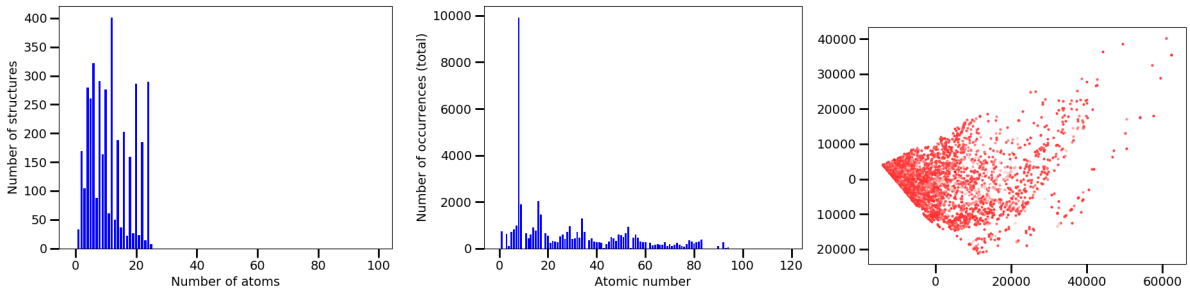


(a) Size distribution (b) Element distribution (c) 2D PCA

Figure 4.1: The QM7 dataset. This dataset has a maximum crystal size of 23 and very few different elements included.

4.2 FLLA

The FLLA set was constructed by more or less adding random crystal structures from the Materials Project (MP) database. It contains 3938 crystal structures and the maximum size of the crystal unit cells is 25 atoms. The minimum formation energy of a crystal in the dataset is −91.41 eV. The size distribution, element distribution, and 2D PCA of FLLA is shown in figure 4.2. From this figure, it can be seen that FLLA has a widespread amount of elements among its structures, with a very large amount of oxygen as the dataset contains a bias towards oxides. This dataset was included to investigate how well the different representations perform with a very diverse dataset. The FLLA set was first used in Ref. [3].



(a) Size distribution (b) Element distribution (c) 2D PCA

Figure 4.2: The FLLA dataset.

4.3 TAATA

The TAATA sets [2], [26] contain few elements, but widely different structures. An important difference between the TAATA set and FLLA or QM7 is that TAATA mostly contains theoretical structures, many of which are not thermodynamically stable. These datasets were constructed in order to create phase diagrams (see chapter 1.2) for these combinations of elements. The estimated error in formation energy when this dataset was constructed was between 0.05 and 0.08 eV/atom. This uncertainty was considered to be on the same level as uncertainties gained through experiments. There are three subsets of TAATA:

- Hf-Zn-N, 4229 structures, maximum crystal size 99 atoms, minimal formation energy −918.69 eV
- Ti-Zn-N, 3775 structures, maximum crystal size 92 atoms, minimal formation energy −814.44 eV
- Zr-Zn-N, 3203 structures, maximum crystal size 48 atoms, minimal formation energy −349.87 eV

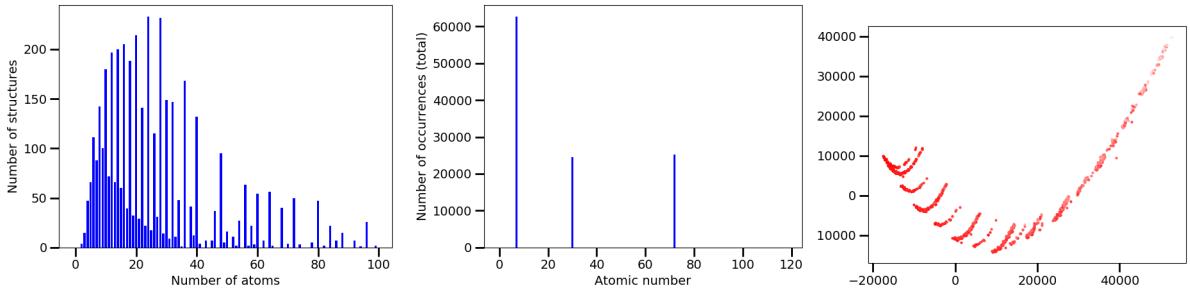
The name of the subset describes the elements that can be found within the dataset. For simplicity, the names will in this report be shortened to *TAATA-Hf*, *TAATA-Ti*, and *TAATA-Zr*. The full dataset will be referred to as *TAATA-Full*. The TAATA set was included in this work to allow an estimation of whether machine learning could have been used to accelerate the creation of the phase diagrams for these elements.

Defective structures

In a late stage of the project, it was discovered that a few of the structures present in the TAATA set are defective. These are three in TAATA-Hf, five in TAATA-Ti, and three more

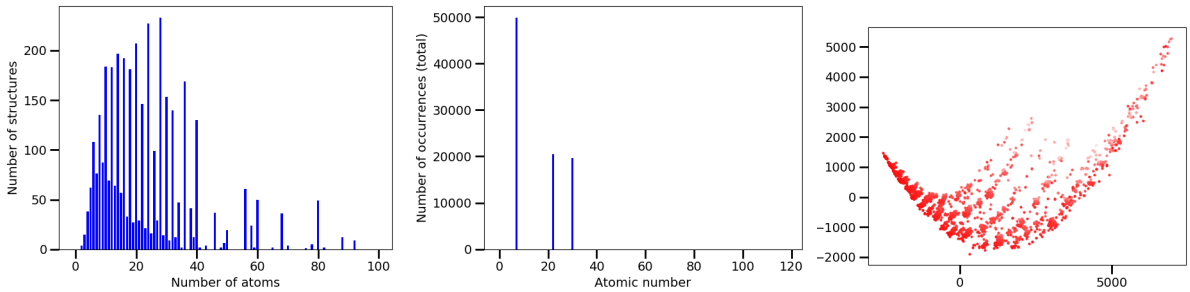
in TAATA-Zr. Some atoms in these structures were placed on top of each other. These defective data were present in the datasets up till that point, and this should be kept in mind when evaluating the results. However, the investigations so far suggest that these have not significantly affected the reported results.

The size distribution, element distribution, and 2D PCA of these datasets are shown in figures 4.3, 4.4, 4.5, and 4.6. The 2D PCAs here do not contain the defective structures. The "full" PCAs can be found in Appendix C. The three subsets are quite similar, containing few elements (nitrogen being the dominant one), a wide range of structure sizes, and PCAs with a "stringy" shape.



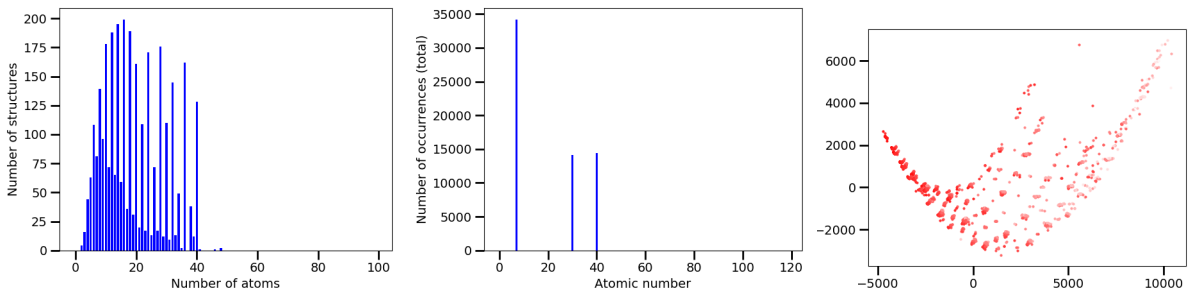
(a) Size distribution (b) Element distribution (c) 2D PCA

Figure 4.3: The TAATA-Hf dataset.



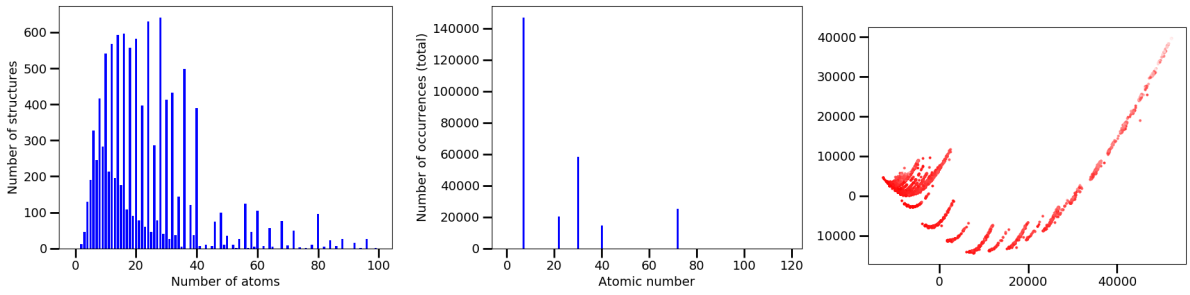
(a) Size distribution (b) Element distribution (c) 2D PCA

Figure 4.4: The TAATA-Ti dataset.



(a) Size distribution (b) Element distribution (c) 2D PCA

Figure 4.5: The TAATA-Zr dataset.



(a) Size distribution (b) Element distribution (c) 2D PCA

Figure 4.6: The TAATA-Full dataset.

4.4 Compactness of Datasets

When speaking about machine learning, and datasets in general, one can talk about the *compactness* of a dataset. In this report, we will use the term compactness to describe how similar the data in a dataset are to each other. Imagine that the data would be plotted in some n-dimensional space where the axes describe different properties of the data. A compact dataset would then have its points clustered together while a dataset which is not compact would have its points widely scattered in the n-dimensional space.

A compact dataset would make it easier for the machine to make accurate predictions since many of the data points and properties will be similar to each other. However, as discussed in chapter 2.4, having a very compact dataset may also cause a severe bias towards this type of data.

To some degree, PCAs (described in chapter 3.5) tell us something about the compactness of a dataset. However, PCA use a specific representation, meaning that the plot is distorted and rescaled by the representation. On top of that, only some properties are present in a PCA, since only some properties are present in the data representation. While the PCA can be used to give hints of the compactness of datasets via the variance of the data, it is by no means an absolute method of determining it.

5 Summary of Machine Learning Scheme

Chapter 2, 3, and 4 have discussed different parts necessary to perform the machine learning. Below is a summary of the machine learning procedure, from raw data to a performance estimation.

1. Transform raw data according to the representation chosen. In some cases, this step also requires matrices to be flattened into vectors, for example when using KRR. The value of the property to be predicted ($\boldsymbol{y}$) should be kept separately.
2. Separate the data into training and test set. If cross-validation is used then this should be done according to the chosen cross-validation scheme.
3. Use the representations ($\boldsymbol{X}_{train}$) and sought properties ($\boldsymbol{y}_{train}$) of the training set to construct the mathematical model for the relationship between these ($f(\boldsymbol{X})$) according to the chosen ML model. In the case of KRR, this means finding the vector $\boldsymbol{\alpha}$.
4. Use the test set to calculate predictions for the sought property: $\boldsymbol{y}_{pred} = \boldsymbol{f}(\boldsymbol{X}_{test})$.
5. Compare the predicted value ($\boldsymbol{y}_{pred}$) to the correct value ($\boldsymbol{y}_{test}$) and calculate the MAE using these.

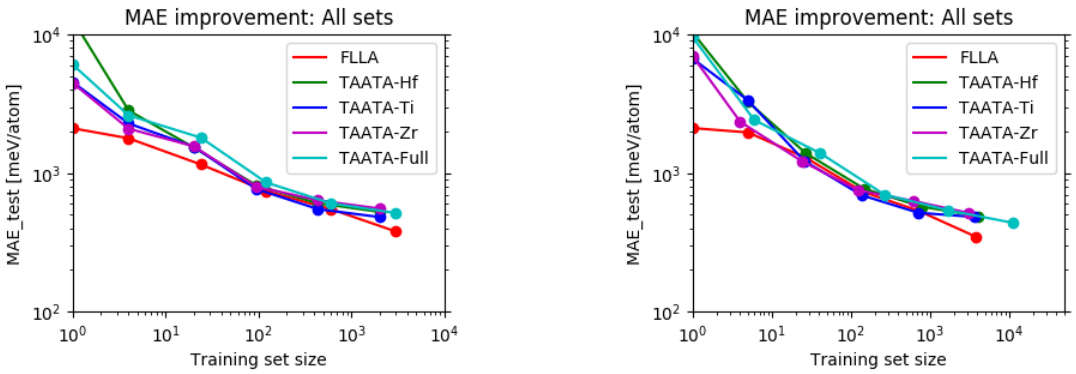
(6. Repeat if cross-validation is used. Then, calculate the mean of the MAE from the different runs.)

6 Results

This chapter presents the results of applying different data representations and other details about the machine learning of the FLLA and TAATA sets.

6.1 Replication of Old Results

The first part of the project was to replicate previous results, primarily the results from the latest project where few elements, but many different crystal structures were used [2]. As a starting point, the old results from Ref. [3] were successfully replicated. These show how the sinusoidal representation performs when applied to the different datasets. The results are presented in figure 6.1.



(a) With the same training set sizes as in Ref. [2].

(b) Extended version.

Figure 6.1: The learning curves for the different datasets using the sinusoidal representation.

When using 3000 training data, taken from the FLLA set, the sinusoidal representation gets an MAE of 0.38 eV/atom (which is approximately the same result as in the original article: Ref. [3]). The MAE becomes 0.52 eV/atom when using 2000 training data from the TAATA-Hf set, 0.48 eV/atom with 2000 training data from the TAATA-Ti set, 0.55 eV/atom with 2000 training data from the TAATA-Zr set, and 0.51 eV/atom using 3000 traing data

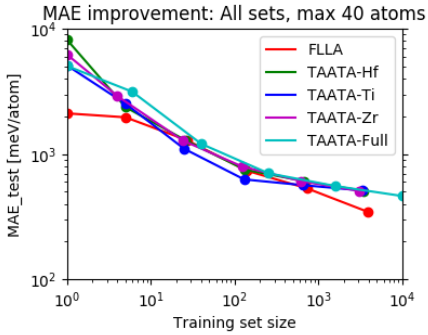


Figure 6.2: The different datasets where all structures which has more than 40 atoms in its unit cell are removed.

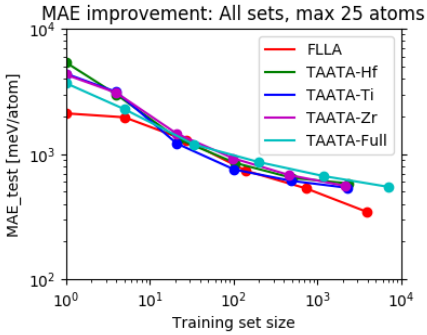


Figure 6.3: The different datasets where all structures which has more than 25 atoms in its unit cell are removed.

from the TAATA-Full set. The results reported in the present work have slightly lower MAE than in the previous work (see figure 2 in Ref. [2]) [1]), where a machine trained on FLLA had an MAE of 0.39 eV/atom, one trained with TAATA-Hf reached down to 0.9 eV/atom, with TAATA-Ti to 0.76 eV/atom, and with TAATA-Full to 0.83 eV/atom for the same training set sizes as above. Unfortunately, no previous MAE result for the TAATA-Zr set is available, although it is included as a subset in TAATA-Full. More details on the results of training and testing with the sinusoidal representation can be found in Appendix B.

Figure 6.1 b) shows the same datasets and implementation, but using larger training set sizes. With the largest training set sizes the resulting MAE are 0.35 eV/atom using FLLA with 3838 training set size, 0.49 eV/atom using TAATA-Hf at 4129 training set size, 0.48 eV/atom with TAATA-Ti and 3674 training set size, 0.52 eV/atom with TAATA-Zr and 3103 training set size, and 0.44 eV/atom using TAATA-Full with 11106 training set size.

Small Datasets

The distribution of the predicted values versus the DFT-calculated values shown in Appendix B suggests that a cut-off based on size was made in Ref. [2] (see discussion in 7.1). That is, large structures (with many atoms) may have been removed from the dataset as they are under-represented in the dataset as a whole. It can be seen in figure 4.6 that some sizes are indeed under-represented. To investigate if such a cut-off was performed in the previous project, two different cut-offs were tested, at 25 and 40 atoms, and these smaller datasets were used for ML training and testing. 40 atoms was a probable cut-off size judging from figures 4.3 - 4.6 a), since all sizes larger than 40 are severely under-represented. 25 atoms was another probable cut-off size as this was the maximum number of atoms in the FLLA set, and the previous project sought to compare performance for the FLLA set with the performance for the TAATA sets.

The results from the machine learning are shown in figure 6.2 and figure 6.3. In figure 6.4, the results for different sizes are summarized for TAATA-Full. In figure 6.2 and figure 6.3 FLLA is also present as a reference line. As FLLA only contains structures with 25 atoms or less, it is not affected by either of the cut-offs. Again, further details of these results are shown in Appendix B, in figure B.2 and B.3.

6.2 Comparison of Representations

Figures 6.5 - 6.7, along with table 6.1 and 6.2 show the result of the tested representations with the different datasets. Figure 6.5 shows that the result is virtually the same, no matter

[1] In this report "Phase 1" corresponds to TAATA-Hf, "Phase 3" to TAATA-Ti, and "Phase All" to TAATA-Full

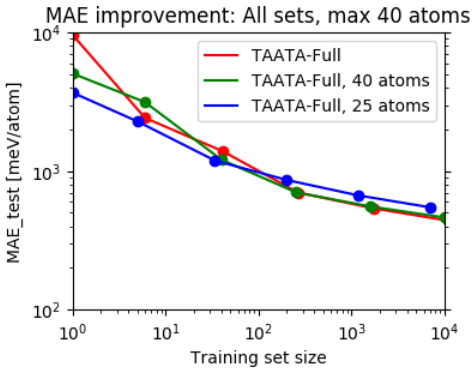


Figure 6.4: The effect of removing structures from the dataset based on unit cell size is summarized using the different plots for TAATA-Full. In this plot, the sinusoidal representation is used.

if additional information is applied to the sinusoidal representation or not. Tables 6.1 and 6.2 show the results of applying the pre-made functions from Ref. [9] to the datasets without any new machine learning. Model 1 gives good results for the FLLA set, but bad results for the TAATA sets. Model 2 is overall much worse than model 1 but otherwise follows the same behaviour as model 1. Figure 6.6 compares the sinusoidal representation and the KRR versions of model 1 and 2. Model 1 gives poor results for the FLLA set, but better for the TAATA sets. The error is on par with the sinusoidal representation, but sometimes a bit worse. Generally, model 2 gives poor results.

Figure 6.7 compares the sinusoidal, PRDF, Voronoi, and SLATM representations with each other. The sinusoidal and PRDF representations in general performs similarly well. In some cases, the sinusoidal representation performs slightly better than PRDF. The SLATM and Voronoi representations perform better than the other two representations. SLATM reaches an end value of 0.42 eV/atom for 11096 training set size with the TAATA-Full set, while the Voronoi representation reaches 0.28 eV/atom at 10086 training set size for the same dataset. The SLATM representation was not applied to the FLLA set, see reasons stated in chapter 7.8. The Voronoi representation gives an MAE of 0.17 eV/atom for FLLA at 3544 training set size.

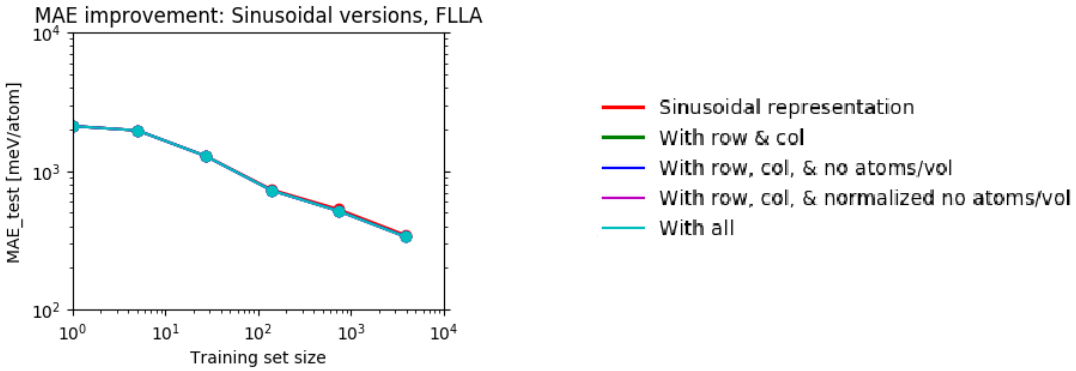
Similar to figure 6.4, figure 6.8 shows that making cut-offs based on unit cell size has no considerable effect on the Voronoi representation either.

Table 6.1: Results from using the first pre-made model.

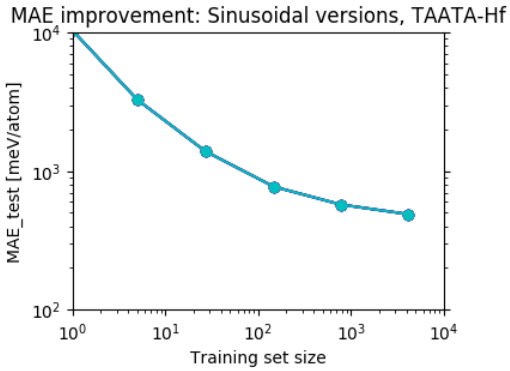
Dataset	Test set size	MAE [eV/atom]
FLLA	769	0.28446
TAATA-Hf	257	6.22054
TAATA-Ti	436	5.64370
TAATA-Zr	224	6.11171
TAATA-Full	917	5.91969

6.3 Lost Data

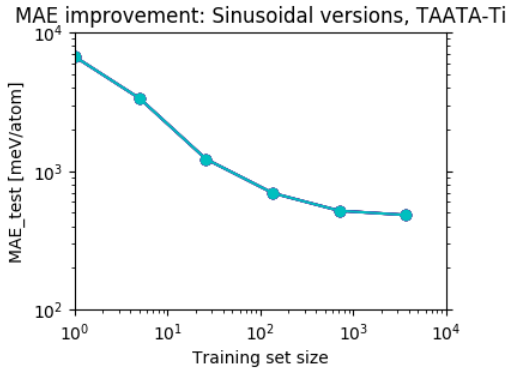
As may have been noticed from the above figures and tables, some of the runs are not performed on the full datasets. The reason for this is individual for each representation. The composition-only representations require very specific properties to be calculated, many of which cannot be determined with the data at hand. Most often, the problematic properties were the formal charges (the charge assigned to an atom in a compound) or the FERE (Fitted



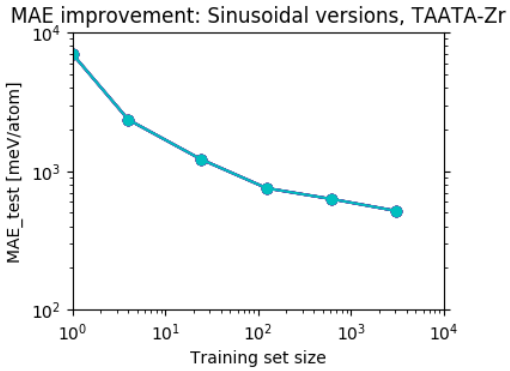
(a) FLLA



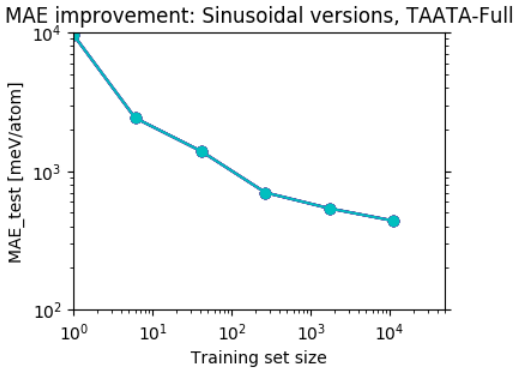
(b) TAATA-Hf



(c) TAATA-Ti

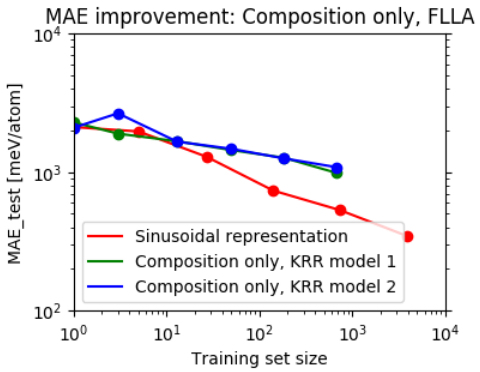


(d) TAATA-Zr

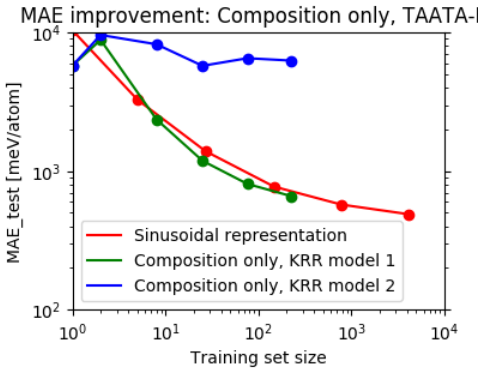


(e) TAATA-Full

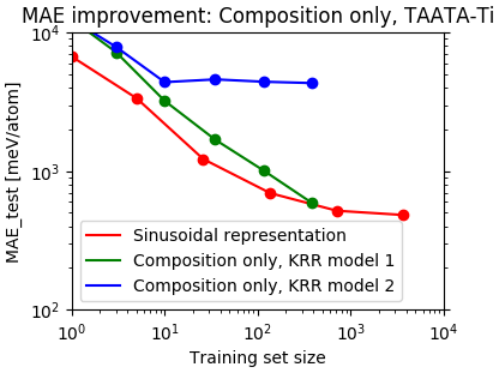
Figure 6.5: The effect of adding information about row, column, and atoms per volume to the sinusoidal representation.



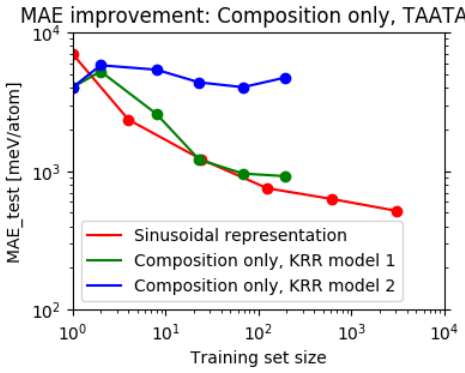
(a) FLLA



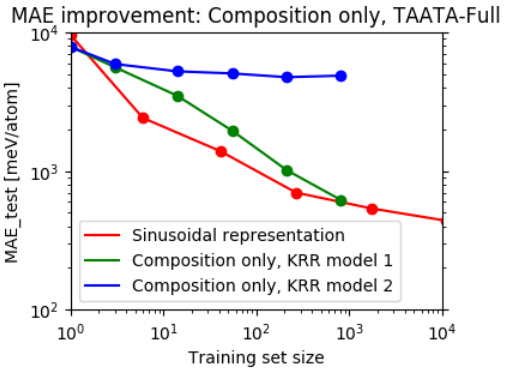
(b) TAATA-Hf



(c) TAATA-Ti

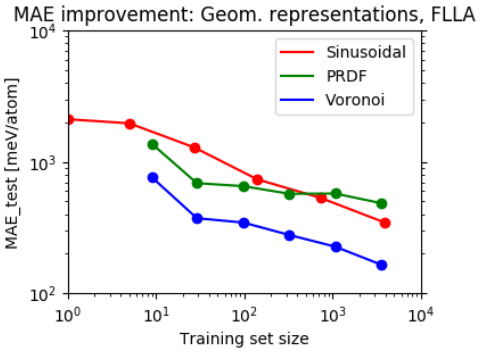


(d) TAATA-Zr

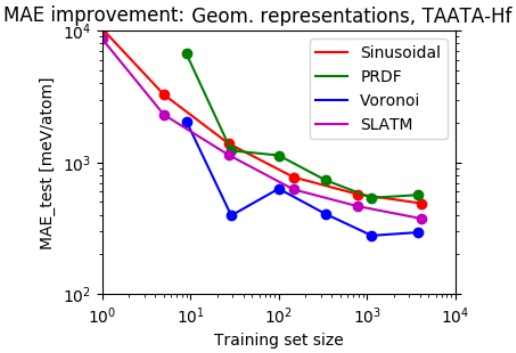


(e) TAATA-Full

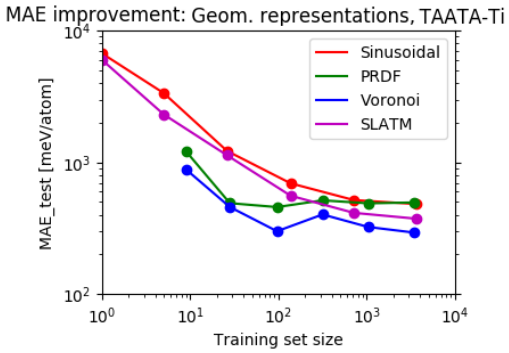
Figure 6.6: The sinusoidal representation compared with the composition-only representation.



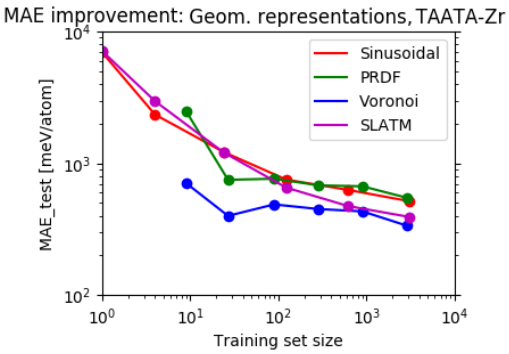
(a) FLLA



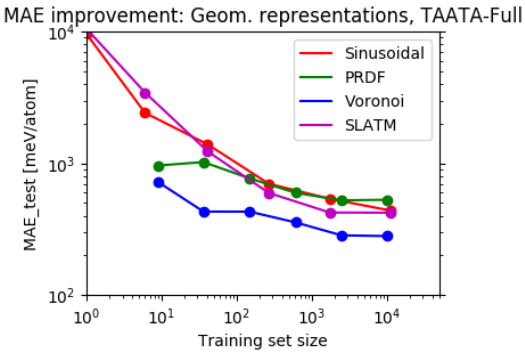
(b) TAATA-Hf



(c) TAATA-Ti



(d) TAATA-Zr



(e) TAATA-Full

Figure 6.7: The sinusoidal, PRDF, Voronoi, and SLATM representation compared with each other.

Table 6.2: Results from using the second pre-made model.

Dataset	Test set size	MAE [eV/atom]
FLLA	776	43.22482
TAATA-Hf	257	502.98161
TAATA-Ti	436	621.43323
TAATA-Zr	224	498.54389
TAATA-Full	917	558.21701

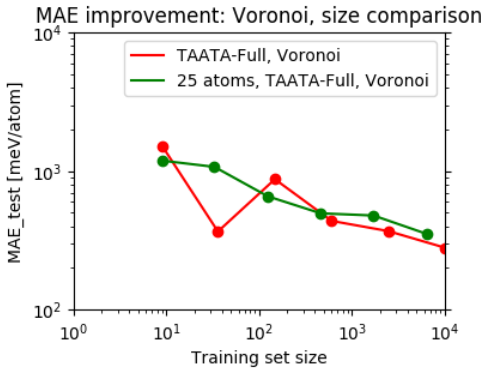


Figure 6.8: The effect of removing structures from the dataset based on unit cell size for TAATA-Full. In this plot, the Voronoi representation is used.

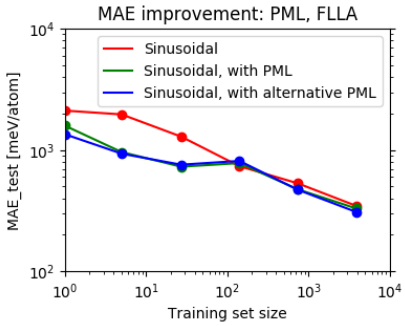
Elemental-phase Reference Energies) values [2]. The formal charges cannot always be determined, as multiple possibilities exist for some elements, and the FERE values are at this point in time only known for some elements. SLATM did not accept the defective data (see chapter 4.3), which eventually led to their discovery. Therefore SLATM was run without these data. How many data that could be used for each representation is reported in table A.1.

6.4 PML

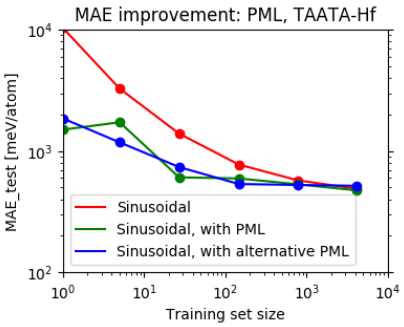
Figure 6.9 shows a comparison of the sinusoidal representation with and without preparatory machine learning. For small training set sizes, the MAE is lower than for the implementation without PML. Eventually, the difference levels out. There is only a small difference between the common PML and the alternative version, and none seems better than the other.

To investigate if this effect was caused not by the PML itself, but the deviation training, the QM7 set was also brought into the picture. QM7 has previously been used to show that PML may be used to lower the error [1]. Figure 6.10 shows a comparison of using the Coulomb representation on the QM7 set with and without deviation training and PML.

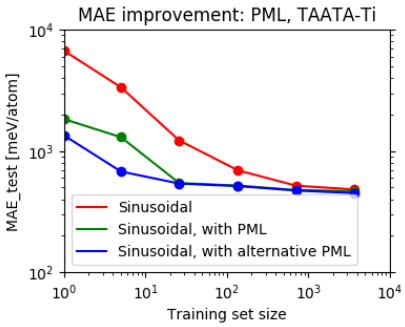
[2] An error correction term to calculations of formation energies of metal-nonmetal compounds. The FERE values are more closely explained in Ref. [27]



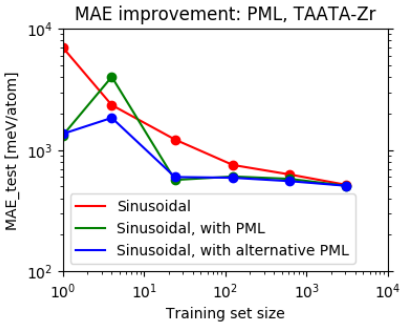
(a) FLLA



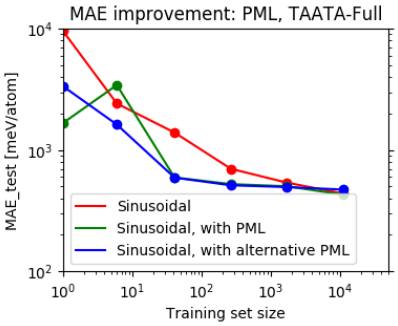
(b) TAATA-Hf



(c) TAATA-Ti

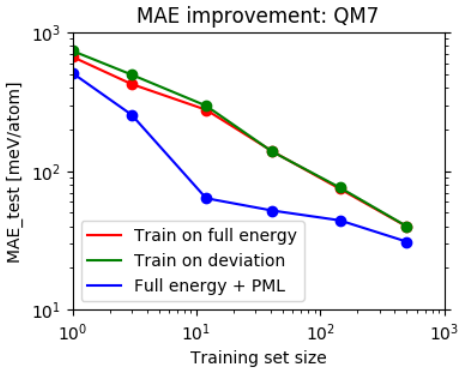


(d) TAATA-Zr

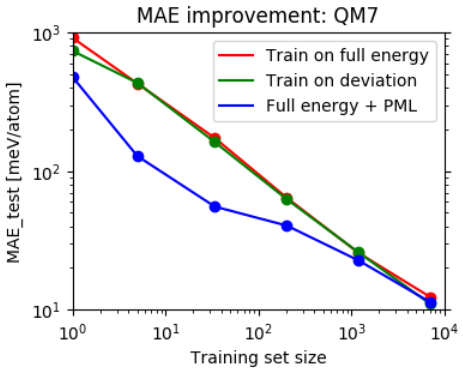


(e) TAATA-Full

Figure 6.9: The effect of PML.



(a) Up to 500 training size.



(b) Up to 7065 training size.

Figure 6.10: The effect of PML on the QM7 set using the Coulomb representation.

7 Discussion

This chapter discusses the results from chapter 6 and the methodology used in this project.

7.1 Reproduction of Old Results

From figure 6.1 it can be determined that the reproduced results generally have a lower MAE than the original results [2]. There will, of course, always be some difference caused by the randomly chosen training and test sets, but these differences are too great and too consistent to simply be a coincidence. The difference is most likely caused by several things.

First of all, it is suspected that different kernel widths have been used. It is likely that the original results used a kernel width of $4 \cdot 10^4$ for all datasets, since this optimum had earlier been used for the FLLA set, and there has been no noted optimum reported for the TAATA set in earlier reports. Secondly, the original work used the total energy of the structures for training, instead of the deviation from the mean, as has been done in this thesis. This is indicated by the scatter plots, figures 3-7 b) in Ref. [2], by noting the "line" of points that sometimes appears at predicted energy value 0. These points are most likely those not caught by the kernel function, and thus returns zero (as explained in chapter 2.5). In the new scatter plots, similar lines should appear at the mean value of the training set, making them more difficult to perceive.

A large difference between the original results and the reproduced ones are the scatter plots of TAATA. The formation energies of the structures does not seem to drop below -250 eV in the original scatter plots, which is unexpected. It is known that all datasets have structures with lower formation energy, see chapter 4. This may indicate that some structures have been removed from the dataset, most likely the larger ones, since these tend to have lower formation energy. Judging from the scatter plots for the reduced datasets (figures B.2 and B.3), it seems probable that a cut-off at 25 atoms has been performed, as the 25-atom cut-off plots are more similar to the original ones.

This cut-off may have been done in an attempt to lower the MAE by creating a more compact dataset. As can be seen in figure 4.3, 4.4, 4.5, and 4.6, very large structures are under-represented in the dataset, so removing the larger ones may indeed create a dataset which is more compact, and thus may be easier to predict. However, figure 6.4 shows that doing a cut-off on this dataset has very little impact on the end result. Figure 6.8 supports this conclusion, as this effect holds for other representations as well.

Therefore it is concluded that the main reason for the difference between the original and the reproduced result is the presumed different kernel width and the deviation training.

7.2 Representations

The second goal of this project was to compare different representations to see which ones are most useful for the construction of phase diagrams in order to determine material stability.

Sinusoidal Representation

Figure 6.5 shows that complementing the sinusoidal representation with information about the row, column, or atoms per volume does not decrease the MAE significantly. This result would indicate that either this information is not relevant for the formation energy, or alternatively, that the information is already covered in the representation in another way. For example, information about the row or column number is connected to the atomic number of the element, which is included in the representation.

Composition Only

Unsurprisingly, the pre-made models (see chapter 3.2) performed quite poorly for most of the data, see table 6.1 and 6.2. The only one that performed well was model 1 with the FLLA set. Both models perform considerably poorer for the TAATA set compared to the FLLA set. It can also be seen that model 2 performs much worse than model 1, regardless of the dataset used. These aspects suggest that the models are not general, but very specialized towards a specific dataset. Model 2 appears to have been constructed for a more compact dataset than model 1, which explains its poor performance when applied to something very different from its original dataset. That model 1 performed considerably well for the FLLA set may indicate that the FLLA set is quite similar in nature to the dataset that the model was originally constructed for. However, since the corresponding KRR version on the FLLA set performed worse than the linear one, this might have just been a coincidence. Otherwise, if the FLLA set was similar to model 1's original dataset, then the KRR would have been expected to perform equally well, if not better.

The KRR version of these models performed considerably better than its pre-made versions, see figure 6.6. In most cases, the composition-only representation gave a larger error than the sinusoidal representation, but the slope of model 1 suggests that with more data, it might be able to give better predictions than the sinusoidal representation. However, more data is required to draw such a conclusion. The risk is that composition only is enough for the machine to learn up to a certain point, where the learning stops and the error curve flattens quickly.

That model 1 performs consistently well for the TAATA set may indicate either that the TAATA set is similar to the dataset which model 1 was constructed for, or that some of the selected properties of model 1 happen to be very central to the formation energy of the TAATA set. Model 1 was constructed for a dataset of compositionally diverse, stable, metal-nonmetal compounds. As the compounds in the TAATA sets are neither compositionally diverse nor restricted to stable compounds, the only common factor is the metal-nonmetal property. The likeliest scenario is that one or several of the properties in model 1 is closely connected to the formation energy of metal-nonmetal compounds.

It should also be noted that applying a pre-made function to a dataset in this manner is not representative of how well the general method described in Ref. [9] performs. Each model built from an ML method is bound to be specialized towards the dataset which it was built for. The above results of applying these models to other datasets are therefore not bad per sé. Rather, they are surprisingly good.

From these two tests, combined with the earlier results presented in Ref. [9], we can draw the conclusion that the method described in Ref. [9] which selects the properties to be used in each model leads to models that are very dataset-specific. Unfortunately, this can make results very hard to use on new data, as a new model needs to be generated for each dataset. If there was some reliable way of knowing the level of similarity between data then this method may become useful. In that case, models made for similar datasets could be applied to new data. However, being strictly composition-based, the representation still possesses the problem of differentiating between crystals with the same composition but different structure.

PRDF, Voronoi, and SLATM

From figure 6.7 it can be concluded that for both the FLLA set and the TAATA set, the Voronoi representation performs best, followed by SLATM, the sinusoidal representation, and lastly PRDF. In the earlier comparison between the sinusoidal, Voronoi, and PRDF representations (figure 3 in Ref. [10]) these show the same behaviour with Voronoi being the best, and PRDF being slightly worse than the sinusoidal representation.

7.3 Effect of PML

The effect of applying PML is significant for small training sets but converges to almost nothing for larger training sets, see figure 6.9. It can be seen that the alternative PML (PML without a constant addition, see chapter 2.5) does not perform better than the normal PML. As PML showed some promise earlier [1], this result is slightly surprising. One suspicion is that the earlier results did not use deviation training and that the supposed benefit of using PML actually was due to PML creating an effect similar to that of deviation training. The benefit of deviation training was to shift the prediction of outliers to the mean instead zero. When performing PML, an additional representation is used, which may cause previous outliers to no longer be outliers, and a large chunk of the formation energy can still be predicted. Even if the non-linear contribution still returns zero, the error of the full prediction will be better than without PML.

In other words, since PML and deviation training share benefits, they converge towards the same result, as in figure 6.9. To confirm this, a test on QM7 with and without PML and deviation training was performed, see figure 6.10. The hypothesis would then be that runs without deviation training would be improved by both PML and deviation training, and that the results using PML and the results using deviation training would converge.

What can be seen in these figures is again that PML improves the result for training on few data, but that the difference disappears for large amounts of data. However, applying deviation training does not seem to improve the error, contradicting part of the hypothesis. The plots show that only PML has an effect, and only at small training set sizes.

This means that either deviation training does not have a significant effect at all, or the QM7 set does not contain many outliers (that is, this result is specific for this dataset). To determine which one of these two statements is true requires further study.

Figure 6.10 a) shows training up to 500 training set size, which was the interval plotted in Ref. [1] when PML was shown to improve the results. At this point, PML does seem to improve the result significantly. However, as figure 6.10 b) shows, this is not true in the long run. Although PML does seem to be useless in the long run, it may still be of use in cases where little data is available.

7.4 Effect of Defective Data

As described in chapter 4.3, 11 of the 11207 data in the TAATA set were defective. Unfortunately, this was discovered rather late in the project, so almost all results were run with

these data. However, the defective data were very few in comparison to the total. As both the machine learning and the MAE only gains a small contribution from each data, the impact a few specific data can have is very limited. If the contribution from these data would have been very large despite this, then the effect would be fast increases or decreases in the error curves when these data switched between training and test set. This effect has not been especially noted except for few training data, where this is a natural behaviour regardless of defective data. Additionally, in figure 6.7 it can be seen that the "SLATM" curve, which was run without these defective data, has a very similar shape to the "Sinusoidal" curve, which was run with these defective data. These arguments indicate that the defective data did not have a large influence over the results. If there is still doubt whether these data distorted the results or not, then the best course of action would be to re-run the results without these data and compare them to the results in this report.

7.5 Bouncing Curves

Generally, the Voronoi and PRDF curves show more irregular behaviour than other representations. The most reasonable explanation is that this behaviour is primarily caused by the cross-validation used (*k*-fold instead of Monte Carlo), and secondly by the data that was randomly selected as training and test set. The PRDF and Voronoi representations were run at the same time, using the same datasets. In figure 6.7 it can be seen that the two curves follow each other, bouncing up and down together. This is especially visible in b) and d). Since the data and cross-validation are the same for both these representations, it is only natural that their curves would have the same behaviour. Additionally, the curves are especially bouncy for small training set sizes, where few data have a large impact on the error. This is even more true for *k*-fold cross-validation since, at these points, *both* the training set and the test set are very small.

7.6 The TAATA Set Versus the FLLA Set

The original thought with the TAATA set was to create a more compact dataset than the FLLA set by constraining the amounts of elements in the dataset. As described in chapter 4.4, a more compact dataset would mean that more structures in the dataset are similar to each other, which should help the machine learning, as this makes it easier to draw upon information from training data to predict test data. However, all the tested representations give an error for the TAATA set equal to or larger than the error for the FLLA set, which indicates that the TAATA set is not more compact than the FLLA set, rather the other way around.

Since the TAATA set is more restricted in its elemental composition, this must mean that it is more diverse in the structural aspect. From figure 6.4 we can conclude that it is not the diversity in size that is the problem.

7.7 General Applicability

More important than discussing which representation that yields the best results for the datasets used in this project is discussing the *generality* of these representations. That is, how well do they perform when applied to all kinds of crystal data? Thankfully, and rather surprisingly, most of the representations presented in this project show a good generality. In many cases, representations are very dataset-specific, but both the sinusoidal, PRDF, and Voronoi have displayed similar behaviour for different datasets (compare with Ref. [10]).

The composition-only models have sometimes performed well, sometimes not. Model 2 has generally performed relatively bad for these datasets, which indicates that it is not general. Model 1 performed much better, but sometimes worse, indicating that it is not either

completely general, but rather is good for certain types of data. Also, since the models are completely composition-based, they can not be used to predict properties of crystals with similar composition, but very different geometrical structures.

The general methodology described in Ref. [9], which gave rise to these models could very well be useful if there existed a good (preferably standard) method for describing the similarity between different types of crystal data. Then a model could be created for each group of data, and the method could be used to identify which model to use for predicting the desired property.

The SLATM representation has performed well for the TAATA sets, but there is little information about how it performs on other datasets. Additionally, it could not be performed on datasets containing many different elements, which limits its usefulness. However, it does have a clear possible use: to create phase diagrams. Whether or not it is generally applicable remains to see, and is a prospect for further work.

7.8 Some Words on Time and Memory Usage

The time and memory usage of different methods are important for the usefulness of these different representations. If machine learning is to be used in the search for new materials, then the methods must be reasonably fast and cannot require too much memory.

Most of the representations presented here were run at a standard stationary computer, with 8 GB RAM, without any sort of parallelization. Training and prediction took around 10-20 minutes for a couple of thousand data. With cross-validation (10 folds) that increases to 1-3 hours. The Voronoi and PRDF representations were run via supercomputers at NSC (National Supercomputer Centre) as at least the Voronoi representation required more than 8 GB memory (but usually less than 16 GB). It is uncertain whether this is due to the representation itself or due to the use of Random Forest as ML method.. PRDF took some extra time to optimize, as it had more hyperparameters to decide, and as the search was performed more thoroughly than for the other representations.

As mentioned before, SLATM was incredibly slow for the FLLA set. This is due to SLATM calculating interactions between different types of atoms. The more atom types it must consider, the more time and memory it consumes. As FLLA contains many different types of atoms, calculating SLATM representations for all the data in the set was estimated to take a full year, using the standard computer mentioned above. Generating SLATM of the TAATA set, however, was a matter of minutes.

The time and memory a representation requires are heavily dependent on how well written and optimized the code is. In this project, the program code used came from wildly different places and people, which is why I will not go into more detail regarding this.

7.9 Method

In this section, some faults with the methodology and the side effects these could have will be discussed.

Effect of Delimitations

In this project, KRR with a Laplacian kernel was chosen as the main ML method. The only other ML method used was RF, for the Voronoi representation. Different representations work better with different ML methods. The Voronoi representation, for instance, has been shown to work better with RF than KRR. SLATM is suspected to work better with a Gaussian kernel than a Laplacian kernel. Choosing a specific ML method favours representations that work well with that method. On the other hand, using many different representations and ML methods together may make it difficult to discern if it is the method or the representation that is responsible for a certain result.

The Voronoi representation used RF because of two reasons: it has been shown to work better with RF, and it was easily accessible with the RF. For most of the representations used in this thesis, it is unknown what ML methods that are optimal for the different representations, and then it matters little what ML method that is tested. However, in this case, it was known that RF had previously performed better than KRR for the Voronoi representation. Using KRR despite knowing that RF was better would not do the Voronoi representation justice. The goal is to find a method that predicts formation energy of crystals as well as possible, not specifically finding a KRR method for that purpose. The best course of action would have been to test the Voronoi representation with the KRR as well, even if only to see that it still performs worse than with the RF. Using the same argument, it is admitted that SLATM should have been tested using a Gaussian kernel as well.

Replicability

The results should be possible to replicate as long as the same datasets are used, and the described process in chapter 2 and 3 are followed. There will always be some small difference due to choice of implementation (numerical differences that propagate through the process) and the randomness of training and test set. Using cross-validation decreases the differences between runs due to random training and test set. The differences should not be large enough to change any conclusions drawn from this report.

Sources

Most sources used in this thesis are published articles. Those that are not published articles are instead either works that are highly relevant to the concerned representations, such as Ref. [2] or Ref. [12], or touch on subjects that are widely known, such as cross-validation [18] or overfitting [14].

Optima

Due to time limits, it was decided to use some shortcuts when searching for the optimal hyperparameter values with respect to how these affect the MAE. Among other things, no cross-validation was used, and hyperparameters were tested one at a time, not simultaneously. Generally, it is best to be thorough when searching for the optimum, because of the risk of finding a local instead of a global optimum. If a false optimum is chosen, then it may have a considerable effect on the validation error. Luckily, in the case with TAATA, one may compare the optima for the different subsets, as they are likely to have similar optima. However, it is not certain that the optima presented in table A.1 are global optima. A defending argument for this method is that the time required to find the optimal hyperparameter values may be the factor that determines whether the machine learning is fast enough to be of use or not. If finding the hyperparameter values takes too much time, then there is no benefit in using machine learning. There is a vital trade-off here between time required and performance.

Another problem regarding optima is the PRDF optima, or rather, the data used to find these. Because of how the PRDF was implemented, each run used completely different data, randomly chosen. This means that the data used to find the optimum was not the same for each hyperparameter value, but also that data used to decide the optimum was also used in the final validation, which is not preferable. Doing this may distort the result favourably. This random selection also made the results from the optima search very difficult to interpret, as the data selected as test set had a large influence over the error. The cross-validation should have counteracted both these issues, but the results were still hard to interpret, so it might not have mitigated the effect enough. As above, the optima found for the PRDF may not be global due to these effects.

Other Notes

As previously discussed, when comparing the effect of a single hyperparameter (such as representation used), it is best to keep all other hyperparameters constant. This has not been done in all cases for different reasons. For example, the PRDF and Voronoi representations used k-fold CV instead of Monte Carlo CV, and it is also uncertain whether these used deviation training or not. Another case is the runs which are done with fewer data than normally, such as in figure 6.6. The composition-only representations used a subset of the normal datasets, which essentially is a completely new dataset. Comparing two different datasets, even though they share many crystals, can affect the conclusions a lot. It would have been better to apply the same data to the different representations, or at least to take note of which type of structures that are not present within the subset.

It also should be noted that the code for the Voronoi representation sometimes yielded errors due to numerical issues. The authors of Ref. [10] have been contacted regarding these errors, and it is believed that they will only have a minor effect on the results, as these errors occur very rarely.

7.10 Summary of the Discussion

In this project, earlier work has been reproduced, with better results than before. Differences are due to different kernel width and deviation training. A number of representations have been compared to find those that show promise in the future application of using machine learning for determining stability of proposed materials. Thus far, the Voronoi representation has performed best, followed by the SLATM representation.

Preparatory machine learning improves the error for small training sets, but the effect diminishes as the training set size increases. There is some doubt whether deviation training actually improves the results or not. Results indicate that the TAATA set is not more compact than the FLLA set.

7.11 Further Work

The SLATM and the Voronoi representations should be further investigated. It would also be of some interest to develop a method of evaluating the similarity between data, and further on, evaluating the compactness of datasets. Such methods would help both with choosing which methods and models to apply to datasets, but also more generally help with the construction of datasets to be used in machine learning. This could also be used to show effectiveness and generality of different ML methods.

Of course, further work should also contain more comparisons of representations and ML methods. It would also be appropriate to put some effort into recording time and memory usage for different methods, as this very much determines the usefulness of the methods.

8 Conclusion

The purpose of the project was to investigate machine learning methods and data representations to be used in the search for new materials. More specifically, the goal was to be able to use these in the construction of phase diagrams in order to estimate the stability of proposed materials. Old results were to be reproduced, and a set of different data representations were to be applied to the FLLA and TAATA sets. The effect of using preparatory machine learning was to be evaluated. The research questions defined in chapter 1.5 were as follows:

1. Does the error indeed become larger for the TAATA set than for the FLLA set using the sinusoidal representation, and if yes, why?

   Conclusion: The error does become slightly larger for the TAATA set than for the FLLA set, but not to the extent reported previously. It is suspected that the TAATA set is slightly less compact than the FLLA set, and therefore harder to predict using machine learning.

2. How do the different representations perform when applied to the FLLA and TAATA sets?

   - How well do they compare to each other overall?
     Conclusion: The Voronoi representation has performed best overall, followed by the SLATM representation.
   - Is there an individual difference in performance between the FLLA set and the TAATA set? If yes, how come?
     Conclusion: The only significant individual difference in performance between the two datasets is for the composition-only model 1, KRR version. It performs better for the TAATA set than for the FLLA set. This may be due to one or several of the properties contained in model 1 being closely connected to the formation energy of the structures within the TAATA set.

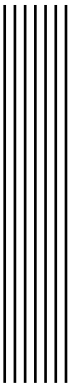
3. Does PML improve the performance of the sinusoidal representation?

   Conclusion: PML improves the performance for small training set sizes, but not for large ones.

4. Does any of the tested representations achieve the goal of an MAE lower than 0.1 eV/atom with a reasonable amount of training data?

   Conclusion: None of the representations achieves the goal of MAE lower than 0.1 eV/atom for the datasets used in this project. The representation which performed best was the Voronoi representation which achieved an MAE of 0.16 eV/atom for FLLA at 3534 training set size and 0.28 eV/atom for TAATA-Full at 10086 training set size.

Since the desired accuracy of 0.1 eV/atom was not achieved, the methods presented in this thesis could not have been used to accelerate the creation of phase diagrams. This result makes it clear that further work is still required if machine learning is to be used in the search for new materials. No combination of ML methods and data representations has thus far been shown to be general and accurate enough to be useful for such a purpose.



Bibliography

[1] F. Faber, "Modeling of crystal formation energies with machine learning", Master's thesis, Linköping University, 2014.

[2] K. Steiner, "Machine learning of formation energies with novel structural descriptors", Linköping University, Tech. Rep., 2017.

[3] 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, Aug. 2015, ISSN: 1097-461X. DOI: 10.1002/qua.24917.

[4] F. A. Faber, A. Lindmaa, O. A. von Lilienfeld, and R. Armiento, "Machine learning energies of 2 million elpasolite ABC_2D_6 crystals", *Phys. Rev. Lett.*, vol. 117, p. 135 502, 13 Sep. 2016. DOI: 10.1103/PhysRevLett.117.135502.

[5] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, "Assessment and validation of machine learning methods for predicting molecular atomization energies", *Journal of Chemical Theory and Computation*, vol. 9, no. 8, pp. 3404–3419, 2013, PMID: 26584096. DOI: 10.1021/ct400195d.

[6] 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, "The Materials Project: A materials genome approach to accelerating materials innovation", *APL Materials*, vol. 1, no. 1, p. 011 002, 2013, ISSN: 2166532X. DOI: 10.1063/1.4812323.

[7] C. Wolverton, J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. 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, 2013, ISSN: 10474838.

[8] S. Kirklin, J. Saal, B. Meredig, A. Thompson, J. Doak, M. Aykol, S. Rühl, and C. Wolverton, "The open quantum materials database (OQMD): Assessing the accuracy of DFT formation energies", *npj Computational Mathematics*, vol. 1, 15010, p. 15 010, Dec. 2015. DOI: 10.1038/npjcompumats.2015.10.

[9] A. Deml, R. O'Hayre, V. Stevanović, and C. Wolverton, "Predicting density functional theory total energies and enthalpies of formation of metal-nonmetal compounds by linear regression.", *Physical Review B*, vol. 93, no. 8, 2016, ISSN: 24699969.

[10] 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: Condensed Matter & Materials Physics*, vol. 96, no. 2, p. 1, 2017, ISSN: 10980121.

[11] K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K. R. Müller, and E. K. U. Gross, "How to represent crystal structures for machine learning: Towards fast prediction of electronic properties", *Phys. Rev. B*, vol. 89, p. 205 118, 20 May 2014. DOI: 10.1103/PhysRevB.89.205118.

[12] B. Huang and O. Anatole von Lilienfeld, "The "DNA" of chemistry: Scalable quantum machine learning with "amons"", *ArXiv e-prints*, Jul. 2017. arXiv: 1707.04146 [physics.chem-ph].

[13] S. Hu, Q. Wang, J. Wang, S. S. M. Chow, and Q. Zou, "Securing fast learning! Ridge regression over encrypted big data", in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 19–26. DOI: 10.1109/TrustCom.2016.0041.

[14] Scikit-learn web page. Underfitting vs. overfitting example, [Online]. Available: http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html (visited on 10/05/2017).

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning : data mining, inference, and prediction.* Ser. Springer series in statistics. New York : Springer, 2009, ISBN: 9780387848570.

[16] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning", *Annals of Statistics*, vol. 36, pp. 1171–1220, 2008. DOI: 10.1214/009053607000000677.

[17] L. Breiman, "Random forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[18] W. Dubitzky, M. Granzow, and D. Berrar, *Fundamentals of data mining in genomics and proteomics.* 2007.

[19] A. Christensen, F. Faber, B. Huang, L. Bratholm, A. Tkatchenko, K. Müller, and O. von Lilienfeld, *Qml: A python toolkit for quantum machine learning.* 2017. [Online]. Available: https://github.com/qmlcode/qml.

[20] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A. V. Lilienfeld, and K.-R. Müller, "Learning invariant representations of molecules for atomization energy prediction", in *Advances in Neural Information Processing Systems*, 2012, pp. 440–448.

[21] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space", *New Journal of Physics*, vol. 15, no. 9, p. 095 003, 2013.

[22] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments", *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, 2016, ISSN: 1364-503X. DOI: 10.1098/rsta.2015.0202.

[23] QM7 Dataset, [Online]. Available: http://quantum-machine.org/datasets/ (visited on 07/14/2017).

[24] L. C. Blum and J.-L. Reymond, "970 million druglike small molecules for virtual screening in the chemical universe database GDB-13", *J. Am. Chem. Soc.*, vol. 131, p. 8732, 2009.

[25] 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, p. 058 301, 2012.

[26] C. Tholander, C. B. A. Andersson, R. Armiento, F. Tasnádi, and B. Alling, "Strong piezoelectric response in stable $TiZnN_2$, $ZrZnN_2$, and $HfZnN_2$ found by ab initio high-throughput approach.", *Journal of Applied Physics*, vol. 120, no. 22, pp. 225102-1 - 225102-6, 2016, ISSN: 00218979.

[27] V. Stevanović, S. Lany, X. Zhang, and A. Zunger, "Correcting density functional theory for accurate predictions of compound enthalpies of formation: Fitted elemental-phase reference energies", *Physical Review B*, vol. 85, no. 11, p. 115 104, 2012.

A Appendix A - Optima

A.1 Finding Optima

To find the optimal hyperparameter values it is generally recommended to use cross-validation, as described in chapter 2.3. However, cross-validation is a very time-consuming process, and in this project, it was decided that time would likely be better spent somewhere else. Because of this, a simpler, but less precise method was used to determine optimal hyperparameter values.

For each individual dataset (TAATA-Hf, TAATA-Ti, etc.) 500 randomly selected data were used as training set and 500 more as test set. Then many different values for σ and λ, the two hyperparameters of KRR, were tested, and the error was recorded. Although it has been shown [5] that one should preferably not test these two individually, but simultaneously, to find the optimum, the two were tested separately with the optimal value for σ being determined first. This was decided both because of the small time budget, and because earlier experiences in the group have shown λ to generally have very little impact on the end result unless it is very wildly varied.

The problems that can arise by using this method is, first of all, that the optimum found is most likely only close to the actual optimum, but not spot-on, since the hyperparameters are varied separately. The second thing to watch out for is the bias due to the test and training set used. Since no cross-validation is applied in this stage, the chosen data may have a large impact on what error the method returns for some given hyperparameter values.

First, σ was tested, for values at least between $3 \cdot 10^4$ and 10^8, with λ set to 10^{-4}. More values were tested only if necessary. The σ that gave the lowest MAE was chosen as the optimal value for that hyperparameter. After that, the σ was kept constant at its optimal value, and λ was varied between 10^{-1} and 10^{-10} and optimized in the same manner.

The QM7 and FLLA sets already had established optima for the Coulomb and sinusoidal representation, respectively [3]. QM7 had $\sigma = 2.5 \cdot 10^3$, $\lambda = 10^{-6}$ and FLLA had $\sigma = 4 \cdot 10^4$, $\lambda = 10^{-4}$.

Particularly Small Datasets

In the cases where the datasets in question were particularly small, so that 1000 data is an unreasonably large part of the dataset, the training and test set sizes were changed. In this case either 100 data or 1/8 of the set size was used as training set size and test set size.

Optima for the PRDF Representation

The PRDF representation has, in addition to σ and λ, also *cut-off distance* and *bin spacing* as hyperparameters in the used implementation. Since more hyperparameters are involved, the risk of choosing wrong optima due to test set bias may be larger. To be on the safe side, when finding optima for the PRDF representation, a 10-fold CV was applied to a 500 data subset of the considered dataset. σ was varied from 10^1 to 10^{10} in logarithmic steps of $10^{0.25}$, λ was varied from 10^{-1} to 10^{-10} in steps of $10^{-0.5}$, the cut-off distance was varied between 1 to 17 Å, in steps of 2, and bin spacing was varied between 2 to 20 Å, in steps of 2. This choice of grid search is similar to that of Ref. [10].

A.2 Dataset Optima

Table A.1 shows the optima that were found using the scheme described in A.1. All results shown in chapter 6 uses these values. The kernel width value marked with * was considered a false optimum after seeing the result of a single run on the dataset. Instead, the corresponding full dataset value of $4 \cdot 10^7$ was used, which also was a close competitor for the optimal value, and which produced better results in the full run. This false optimum was most likely an artefact of the test and training set bias, as explained in chapter A.1. The complemented sinusoidal representation used the same optima as the normal sinusoidal representation.

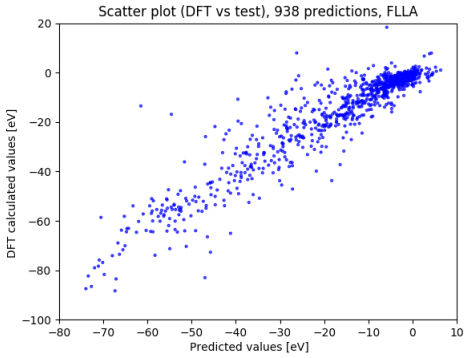
Table A.1: Optima for the different representations and datasets.

Representation	Dataset	Notes	Set size	σ	λ	Cut-off distance	Bin spacing
Coulomb	QM7	-	7165	$2.5 \cdot 10^3$	10^{-6}	-	-
Sinusoidal	FLLA	-	3938	$4 \cdot 10^4$	10^{-4}	-	-
Sinusoidal	TAATA-Hf	-	4229	$3.5 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Hf	max 40 atoms	3578	$4.8 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Hf	max 25 atoms	2468	$6.5 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Ti	-	3775	$5 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Ti	max 40 atoms	3448	$1.7 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Ti	max 25 atoms	2358	$5 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Zr	-	3203	$4 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Zr	max 40 atoms	3199	$9 \cdot 10^4$ *	10^{-4}	-	-
Sinusoidal	TAATA-Zr	max 25 atoms	2254	$1 \cdot 10^8$	10^{-4}	-	-
Sinusoidal	TAATA-Full	-	11207	$8 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Full	max 40 atoms	10225	$6 \cdot 10^7$	10^{-4}	-	-
Sinusoidal	TAATA-Full	max 25 atoms	7080	$1.5 \cdot 10^7$	10^{-4}	-	-
Composition only, method 1	FLLA	KRR version	769	$1.3 \cdot 10^6$	10^{-4}	-	-

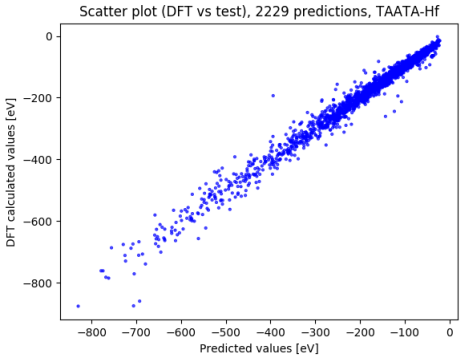
Composition only, method 1	TAATA-Hf	KRR version	257	35	10^{-4}	-	-
Composition only, method 1	TAATA-Ti	KRR version	436	45	10^{-4}	-	-
Composition only, method 1	TAATA-Zr	KRR version	224	10	10^{-4}	-	-
Composition only, method 1	TAATA-Full	KRR version	917	45	10^{-4}	-	-
Composition only, method 2	FLLA	KRR version	776	$1.4 \cdot 10^{6}$	10^{-7}	-	-
Composition only, method 2	TAATA-Hf	KRR version	257	$6.9 \cdot 10^{-}4$	10^{-6}	-	-
Composition only, method 2	TAATA-Ti	KRR version	436	$1 \cdot 10^{4}$	10^{-6}	-	-
Composition only, method 2	TAATA-Zr	KRR version	224	$1.9 \cdot 10^{4}$	10^{-1}	-	-
Composition only, method 2	TAATA-Full	KRR version	917	44	10^{-7}	-	-
PRDF	FLLA	-	3938	10^{1}	10^{-3}	7	7
PRDF	TAATA-Hf	-	4229	10^{2}	10^{-10}	5	2
PRDF	TAATA-Ti	-	3775	$10^{1.75}$	$10^{-8.5}$	11	2
PRDF	TAATA-Zr	-	3203	$10^{0.75}$	$10^{-7.5}$	21	2
PRDF	TAATA-Full	-	11207	$10^{0.5}$	$10^{-7.5}$	13	1
SLATM	TAATA-Hf	-	4226	$1.5 \cdot 10^{6}$	10^{-4}	-	-
SLATM	TAATA-Ti	-	3770	$8.5 \cdot 10^{5}$	10^{-4}	-	-
SLATM	TAATA-Zr	-	3200	$1.7 \cdot 10^{6}$	10^{-4}	-	-
SLATM	TAATA-Full	-	11196	$2.5 \cdot 10^{6}$	10^{-4}	-	-

B Appendix B - Scatter Plots

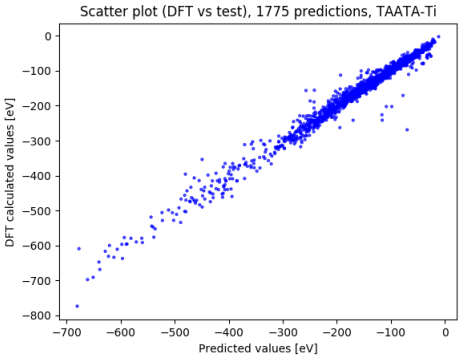
This appendix contains some scatter plots for the sinusoidal representation. The x-axis shows the predicted energy for the test set at hand, while the y-axis shows the DFT-calculated value for the crystals. In the ideal case, the plot should resemble a straight line, i. e. the predicted value is the DFT-calculated value.



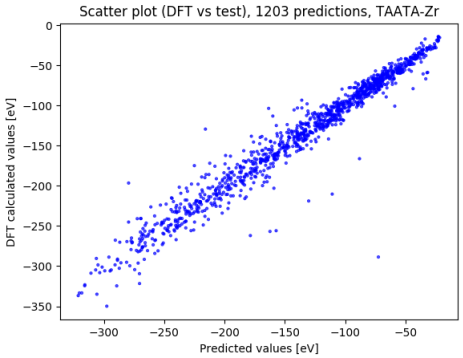
(a) Scatter plot for FLLA, using the sinusoidal representation. Training set size 3000.



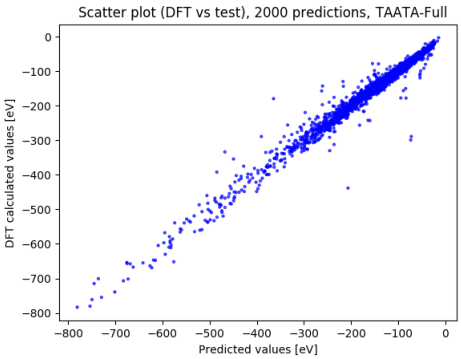
(b) Scatter plot for TAATA-Hf, using the sinusoidal representation. Training set size 2000.



(c) Scatter plot for TAATA-Ti, using the sinusoidal representation. Training set size 2000.

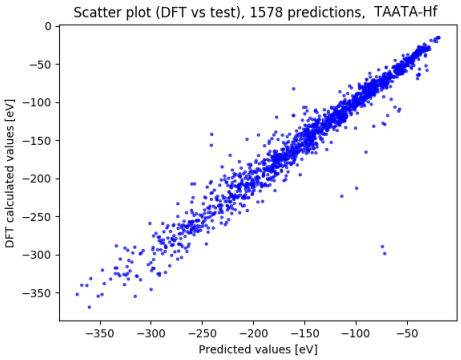


(d) Scatter plot for TAATA-Zr, using the sinusoidal representation. Training set size 2000.

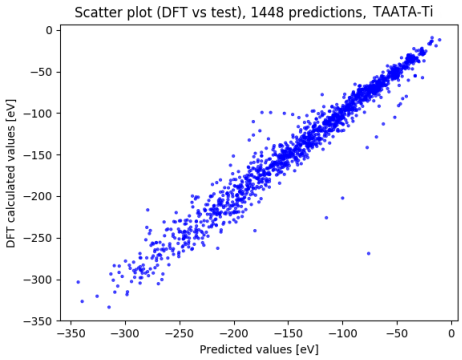


(e) Scatter plot for TAATA-Full, using the sinusoidal representation. Training set size 3000.

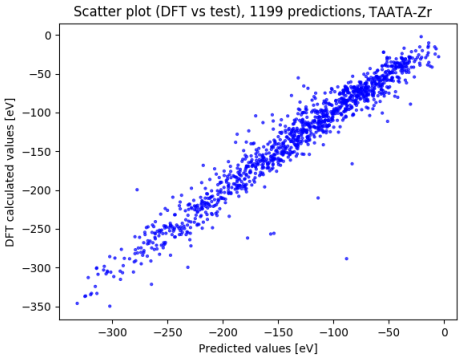
Figure B.1: Scatter plots of FLLA and TAATA.



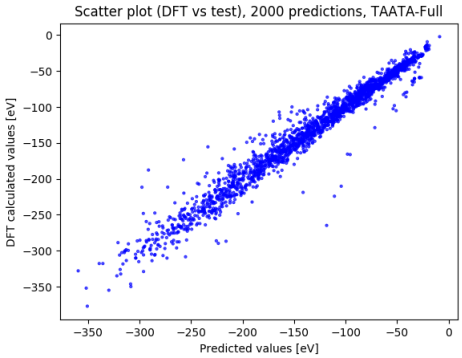
(a) Scatter plot of TAATA-Hf, max 40 atoms. Training set size 2000.



(b) Scatter plot of TAATA-Ti, max 40 atoms. Training set size 2000.

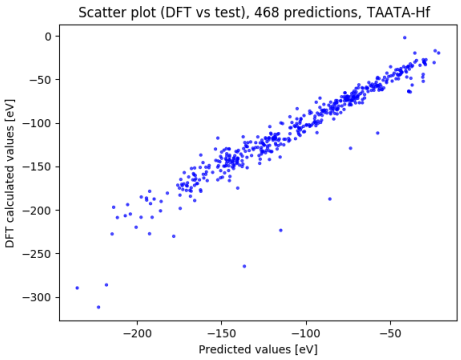


(c) Scatter plot of TAATA-Zr, max 40 atoms. Training set size 2000.

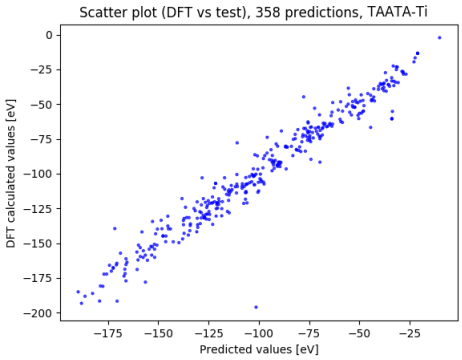


(d) Scatter plot of TAATA-Full, max 40 atoms. Training set size 3000.

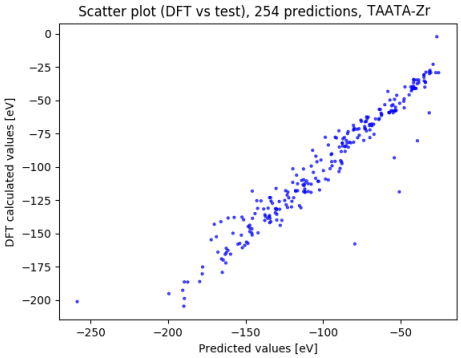
Figure B.2: Scatter plots of TAATA, max 40 atoms in each unit cell.



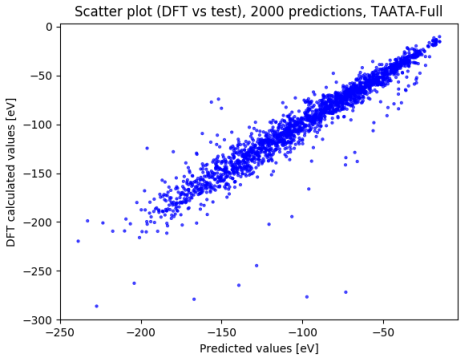
(a) Scatter plot of TAATA-Hf, max 25 atoms. Training set size 2000.



(b) Scatter plot of TAATA-Ti, max 25 atoms. Training set size 2000.



(c) Scatter plot of TAATA-Zr, max 25 atoms. Training set size 2000.

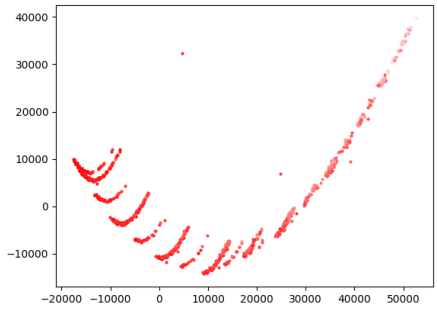


(d) Scatter plot of TAATA-Full, max 25 atoms. Training set size 3000.

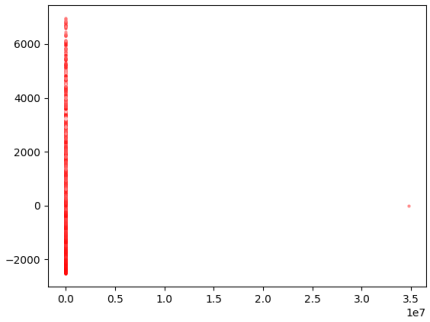
Figure B.3: Scatter plots of TAATA, max 25 atoms in each unit cell.

C Appendix C - PCAs Containing Defective Data

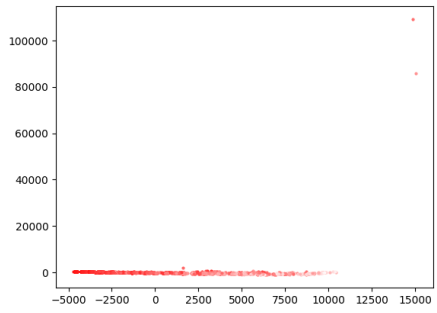
This appendix contains PCA plots of the TAATA sets containing defective data. As can be seen in figure C.1, some of these data seriously distort the PCAs. Especially one point in TAATA-Ti is plotted very far away from the other points.



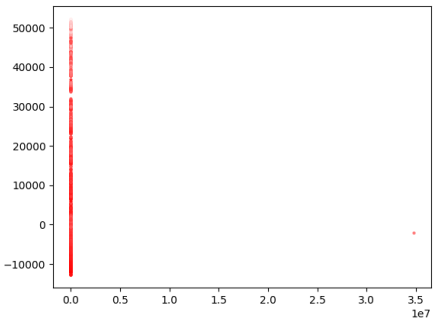
(a) TAATA-Hf.



(b) TAATA-Ti.



(c) TAATA-Zr.



(d) TAATA-Full.

Figure C.1: PCAs of the TAATA sets, using the sinusoidal representation.