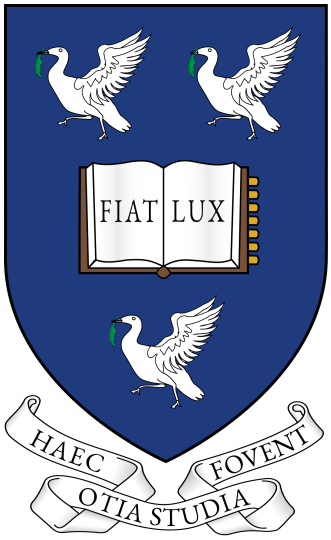
Crystal Energy Prediction with Isometry Invariants of Periodic Point Sets

by **Jakob Ropers** (201183851)

Supervisor: Dr. Vitaliy Kurlin
Secondary Supervisor: Dr. Olga Anosova

Computer Science and Artificial Intelligence
Department of Computer Science
University of Liverpool

May, 2021



Acknowledgements

I would like to thank Dr. Kurlin and the Data Science Theory and Applications research group for enabling me to take on a challenging and relevant problem and supporting me throughout the course of this project. I would like to express special gratitude for Philip Smith, who guided me through the process and background knowledge, allowed close collaboration during this work, and helped me in the writing process of this dissertation. In addition, I want to acknowledge Klara Adams for proofreading my work and helping with wording and grammar in the final project report. Finally, Cameron Hargreaves' help on the use of the Barkla computing cluster was greatly appreciated.

Statement of Originality

This dissertation is submitted as part requirement for the degree "BSc Computer Science with Artificial Intelligence" at the University of Liverpool. I certify that the content of this paper is the product of my own work and all assistance and sources have been acknowledged. I hereby give the permission to loan out copies of this report to future students.

Abstract

Most commonly, pharmaceuticals come in the form of crystalline, rather than amorphous solids, offering greater chemical stability. However, due to the polymorphous nature of some compounds, an identical composition of atoms or molecules can exist in several different crystalline arrangements. These can potentially have different properties, such as the dissolution rate and hence bioavailability. Therefore, a method of predicting polymorphs of a crystalline solid, as well as discovering new crystals of an existing compound, is significant in not only the pharmaceutical industry, but also the discovery and design of organic semiconductors, explosives, and the food industry. The problem of predicting crystal structures on the atomic level is considered a fundamental challenge in material sciences. An important aspect of this process is the crystal energy ranking of the predicted crystal structures, of which only a few are synthesizable and observable, as low crystal energies indicate the thermodynamically feasible and, therefore, stable arrangements.
This work focuses on the crystal energy prediction of crystal structures, exclusively using the geometry of the structure rather than its chemical properties as input features. The geometric structure, represented as a periodic point set, is described using the average minimum distances and density functions; proven stable isometry invariants of periodic point sets. Previous state-of-the-art approaches to energy prediction consider effects of many-body dispersion, exact exchange, and vibrational free energies, delivering relatively accurate predictions at the cost of an extremely computationally intensive algorithm. The primary goal of this work is to use isometry invariants in order to propose a prediction method with much lower computational cost. A range of preprocessing techniques and machine learning methods is explored, evaluated, and critically compared using the T2 dataset, as introduced by Angeles Pulido et al, consisting of labled predicted crystal structures for the Benzimidazolone Triptycene compound.
Although the industry goal of sub-$kJmol^{-1}$ predictions and the project target of predictions with an error of less than 4 $kJmol^{-1}$ is not achieved, failing to outperform current state-of-the-art methods in terms of accuracy, the Gaussian process, using average minimum distances of crystal structures, proves to be a viable approach. This notion is reinforced by its adequate prediction accuracy at a low computational cost, with prediction speeds significantly faster than the current state-of-the-art DFT+MBD approach. This project is a proof of concept for crystal energy prediction, solely using isometry invariants, at computational speeds of only a small fraction of existing state-of-the-art methods and further emphasizes the advantages of statistical machine learning methods in crystal energy prediction.

Contents

List of Figures

List of Tables

Table of Abbreviations

AMD . . . . . Average Minimum Distance
CSP . . . . . Crystal Structure Prediction
DF . . . . . Density Function
MAE . . . . . Mean Absolute Error
MAPE . . . . . Mean Absolute Percentage Error
RMSE . . . . . Root Mean Squared Error
T2L . . . . . Variant of Average Minimum Distances that considers all atoms
T2L-C . . . . . Variant of Density Functions that ignores the three oxygen atoms
T2L-CO . . . . . Variant of Density Functions that considers all atoms
T2L-CON . Variant of Average Minimum Distances that ignores hydrogen atoms

1 Introduction

1.1 Motivation

Most commonly, pharmaceuticals come in the form of crystalline, rather than amorphous solids. The reason for this is that while amorphous solids may have some desirable advantages in comparison to the crystalline alternative, they lack chemical stability and tend to crystallize, potentially changing the effects of the drug [1]. Although crystalline solids are often more stable, it is important to understand their polymorphous nature, allowing an identical composition of atoms or molecules to exist in several different arrangements. Polymorphism is common, occurring in "at least 50% of molecules that have been subjected to industrial polymorph screens" [2]. The various polymorphs of a compound may have different properties, such as the solubility or dissolution rate and hence bioavailability [2, 3]. Therefore, it is important to be aware of the possible polymorphs of a compound as well as the most stable arrangement in order to ensure the desired properties [3]. In fact, legislation on pharmaceuticals, such as the "Guideline on the Chemistry on Active Substances" by the European Medicines Agency[1], requires producers of pharmaceuticals to acquire knowledge about the potential solid forms of an active drug [4]. The use of reliable computational predictions are significant for the design of active pharmaceutical ingredients, as well as the prevention of undesirable change of form during the development, potentially resulting in unwanted properties [5]. Although polymorphs can be discovered through experimental solid form screening, this process is typically on the timescale of months and, therefore, very expensive [6]. Due to this extremely time-consuming and expensive nature of the pharmaceutical industry, an accurate and efficient computational method of discovering new stable crystal structures of a compound and all its existing polymorphs can significantly reduce the cost and timeline of the development of drugs [7]. Similarly, the understanding and efficient exploration of crystalline solids does not only play an important role in pharmaceuticals, but also in the discovery and design of organic semiconductors, explosives, and the food industry [8, 9].
The process of Crystal Structure Prediction (CSP) aims to estimate stable crystal structures based on a compound's molecular formula [10]. The ultimate goal of crystal structure prediction is the exploration of polymorphs, co-crystals, salts, etc. of a compound, solely based on its chemical composition [10]. The problem of predicting crystal structures on the atomic level is considered a fundamental challenge in condensed matter sciences [11]. Although the scientific community has made numerous attempts, some of which are somewhat successful, it has not been able to accept an absolute solution, broadly applicable on a routine industrial scale, since the problem's introduction in the 1950s [12, 13]. However, crystal structure prediction is currently transitioning from exclusively an area of research into an applied technology, making it a topic with significant relevance [6].

[1] www.ema.europa.eu/en/chemistry-active-substances-chemistry-new-active-substances

1.2 Crystal Structures and their Representation

A solid periodic crystalline material, crystal for short, is a solid whose internal order is determined by the three-dimensional periodic arrangement of atoms, molecules, or ions [14]. Most commonly, it is represented by its unit cell and motif, which make up the elementary building block that is periodically repeated in three linearly independent directions [14]. Figure 1 portrays a T2 crystal, represented by its monoclinic unit cell that can be periodically arranged to recreate the entire crystal structure.

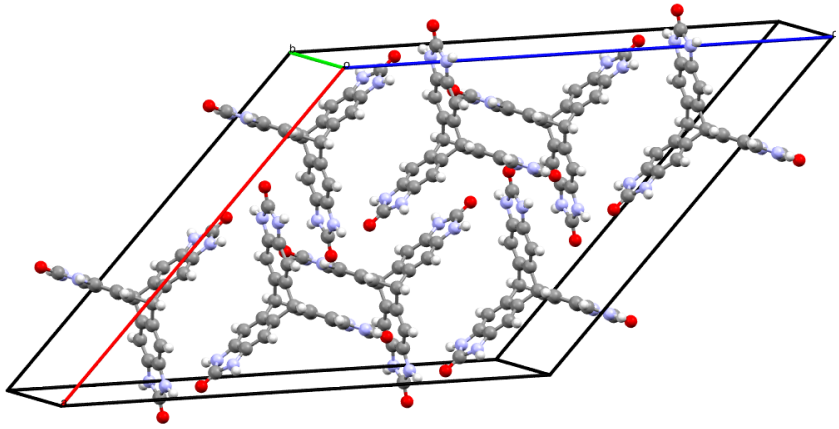


Figure 1: A molecular crystal of the T2 dataset with a monoclinic unit cell. [Visualized with Mercury]

The periodicity of the crystal structure is often represented as a Bravais lattice, which is an infinite "collection of vectors expressed as a linear combination of n linearly independent vectors", where n is the number of dimensions in the space [15]. Formally, a lattice Λ can be defined as $\Lambda = \mathbf{p} + \{\sum_{i=1}^{n} \lambda_i \mathbf{v}_i \ : \ \lambda_i \in \mathbb{Z}\}$, given vector $\mathbf{p}$, spanning from origin of $\mathbb{R}^n$ to any point $p \in \mathbb{R}^n$, and linear basis $\mathbf{v}_1, ..., \mathbf{v}_n$ in $\mathbb{R}^n$ [16]. The Bravais lattice is the combination of the seven crystal systems and their lattice types, resulting in a total of 14 variations [14]. As a lattice is infinite, it can be more useful to only present a small portion of it. The unit cell, defined as the smallest repeating unit which shows the full symmetry of the crystal lattice, is a parallelepiped spanned by its n linear basis vectors [14]. Formally, a unit cell U is defined as $U(\mathbf{v}_1, ..., \mathbf{v}_i) = \{\sum_{i=1}^{n} \lambda_i \mathbf{v}_i \ : \ \lambda_i \in [0, 1)\}$ [16]. The range of possible shapes of the unit cell in a three-dimensional crystal structure is summarized by the seven crystal systems, which include cubic, tetragonal, orthohombic, monoclinic, triclinic, hexagonal, and trigonal unit cells [14]. These crystal systems are governed by the presence or absence of various symmetries, often referred to as the Hermann-Mauguin system [14]. They can be further broken down into 230 space groups, a grouping of specific symmetry in Euclidean space [17]. Finally, the unit cell is populated by the motif, which includes the actual finite set of atoms or ions that make up the elementary building block of the crystal structure. Formally, a motif M is any finite set of points $p_1, ..., p_m \in U$ [16]. Since atoms can be modeled as points of a given radius, centered at the atomic centres, a crystal structure can be mathematically represented as a periodic point set [18]. Formally, the periodic point set is

the Minkwoski sum of the crystal lattice Λ and the motif M, given as $S = \Lambda + M$ [16].

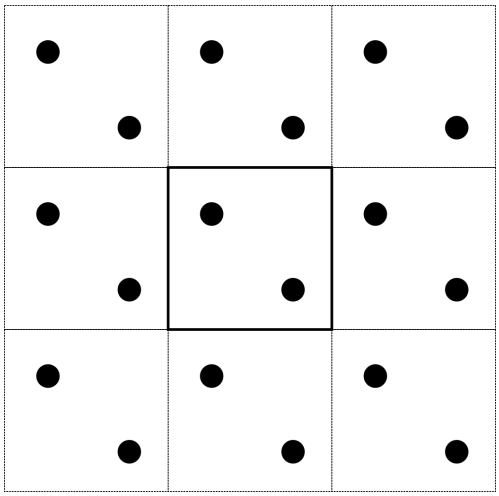


Figure 2: A 2D perspective on a hypothetical crystal lattice, represented by its cubic unit cell and motif. [Visualized with Inkscape]

Figure 2 shows a two-dimensional perspective on a hypothetical crystal structure. The symmetry of the lattice can be described by a cubic unit cell, defined by its three equal side lengths, spanned perpendicular to each other. The motif, populating this unit cell, includes 2 atoms. This unit cell makes up the elementary building block that can be periodically arranged in all three dimensions to recreate the crystal structure.

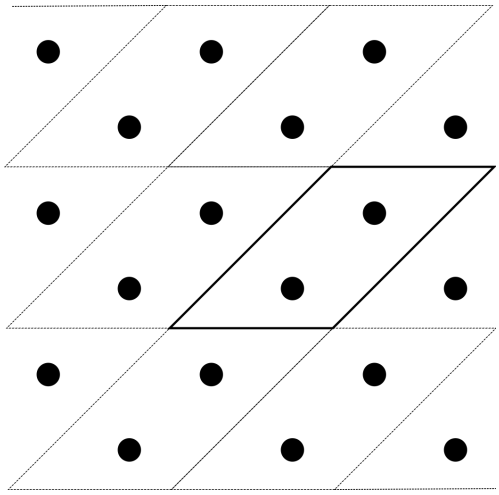


Figure 3: A 2D perspective on a hypotheitcal crystal lattice, represented by its monoclinic unit cell and motif. [Visualized with Inkscape]

To summarize, the use of this description significantly simplifies the representation of the overall structure. Instead of defining a coordinate system and listing all existing atoms or ions and their corresponding coordinates, the breakdown into the unit cell and its motif reduce the complexity of the representation while avoiding

loss of information on the crystal. However, this representation is highly ambiguous, making it extremely difficult when comparing crystal structures for equivalence [18]. With an infinite choice of basis and the corresponding use of different motifs, the same crystal structure can have an infinite number of representations [18]. This means that two crystals with different unit cells and motif may in fact be equivalent, just represented differently. This problem is portrayed in Figure 3, which is a representation of the same hypothetical crystal structure as in Figure 2, but using a monoclinic unit cell. This ambiguity makes the process of CSP, specifically crystal energy ranking, a difficult task as there is no real measure of equivalence. As a result, two similar crystal structures could potentially be considered dissimilar according to traditional features such as their symmetries or unit cells and motif [19].

1.3 Isometry Invariants: Geometric fingerprints of Crystal Structures

The problem of equivalence can be solved with the use of invariants. An invariant is a quantity that is preserved by the equivalence relation [20]. Therefore, two equivalent objects, in this case crystal structures, must hold the same value for the invariant [20]. However, when talking about the opposite, it is important to note that if two objects hold the same value for an invariant then one cannot conclude that they are equivalent. To make this conclusion, the invariant must be complete, meaning that two dissimilar objects must also hold different values for this complete invariant. By applying this completeness, the equivalence problem is solved as two objects that hold the same value for a complete invariant can be assumed equivalent. The use of a representation of crystal structures that is isometry invariant, meaning that it stays constant through distance-preserving transformations and stable under perturbations, is extremely beneficial and is the focus of this paper [19].

1.3.1 Density Functions: Stable Isometry Invariants of Periodic Sets

In the scope of this project, one of the considered isometry invariants for crystal structures, represented as periodic point sets, is the set of density functions (DF). More specifically, these DFs are defined in the paper "The Density Fingerprint of a Periodic Point Set", which introduces the set of DFs as a proven stable isometry invariant of point sets and conjectures their completeness [19]. As this invariant is applied to algebraic geometry in crystal structures, it will be referred to as a geometric invariant. It is important to note that as the crystal structure is treated as a periodic point set information about the chemical elements that make up this structure is ignored.

Formally, given a point set S, made up of its lattice Λ and motif M in $\mathbb{R}^3$, $S(r)$ can be defined as the collection of balls with radius $r \geq 0$, centered at the points in S. The k^{th} DF ψ_k^S of S is the fractional volume of the k-fold cover, meaning the volume covered by exactly k balls with radius r [19]. As shown in Figure 4, as the radius of the balls centered at each point in the square lattice increases, a certain area is covered by zero to four overlaps of the circles. The first density function ψ_1^S shows the percentage area of the total observed area that is covered by one layer

and, similarly, the k^{th} density function ψ_k^S shows the percentage area covered by k layers of the circles. However, this is a simplification in $\mathbb{R}^2$ and when applied to crystal structures, the overlapping volume, rather than area, is calculated in $\mathbb{R}^3$.

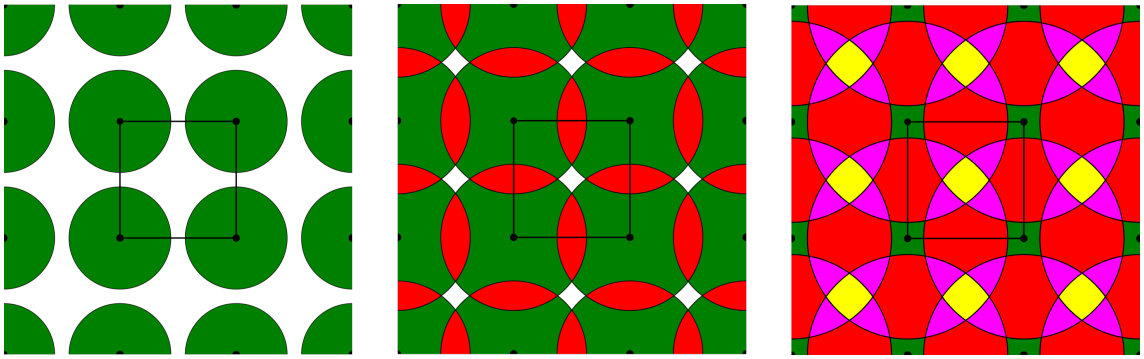


Figure 4: A simple square lattice with circles of growing radius around each point. The density functions express the increase and decrease of these overlapping areas as a representation of the geometric structure. *Left:* There is no overlapping of the circles, marked in green. *Center:* Circles are starting to overlap once, as indicated by the red area. *Right:* Circles are starting to overlap three and four times, indicated by the pink and yellow areas, respectively. [Visualized with Inkscape]

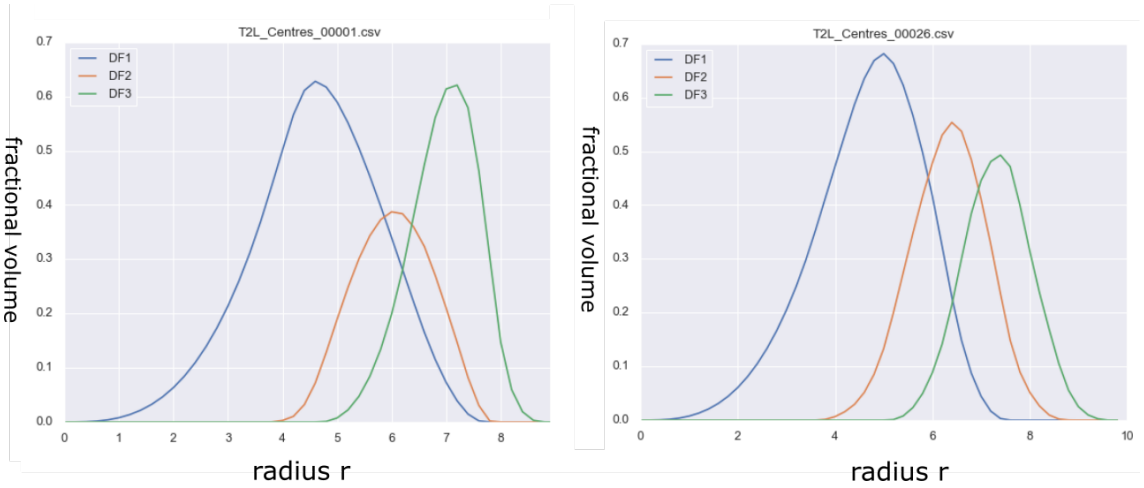


Figure 5: The first three density functions of Job 00001 and 00026 of the T2 crystal dataset. [Visualized with Matplotlib]

The set of DFs is very descriptive and can vary significantly between crystal structures that are not equivalent. For example, Figure 5 shows the first three density functions of job 00001 and 00026 of the T2 crystal dataset. Simply by looking at the two graphs, an observer can objectively claim a difference between the behaviour of the DFs. As the set of DFs is proven to be a stable invariant, the assumption that the DFs of two equivalent crystal structures are identical can be made. However, if two crystal structures have identical DFS, one cannot assume that they are equivalent as the set of DFs is not proven to be a complete invariant.

When considering the balance between computational complexity and information gain, the choice of how many DFs are used is important. Up to a certain threshold, an increase in the computed DFs leads to information gain, although the computational cost also becomes greater. For the purpose of this project, the first 8 DFs of each crystal structure will be considered, measuring the fractional volume of up to the 8-fold cover. This value is suggested by Philip Smith, one of the authors of the original paper, as an optimal balance between information gain and complexity, both computationally and data sizewise [19]. Using more than 8 DFs does not add a significant amount of information but increases the computing time significantly. The computation of the 8 DFs is still costly and takes a significant amount of time on a standard commercial computer.

1.3.2 Average Minimum Distances: Isometry Invariants of Periodic Point Sets

The second isometry invariant, considered in this paper, is the set of Average Minimum Distances (AMDs) of a crystal structure, represented as a periodic point set. The set of AMDs is introduced in the paper "Average Minimum Distances of periodic point sets" and is proven to be a continuous isometry invariant [18]. Although most likely incomplete, AMDs provide a detailed and simple abstraction of crystal structures and can be extremely useful for crystal energy prediction.
In order to formalize AMDs mathematically, one must first understand the Pointwise Distribution of Distances (PDD). Similar to density functions, the crystal structure is represented as a periodic point set S. The PDD of S is defined as the $m \times k$ $matrix$ in which the "i-th row consists of the ordered Euclidean distances $d_{i1} \leq ... \leq d_{ik}$ from the point p_i to its first k nearest neighbours" in the point set S [18]. Based on this, the AMD_k is defined as the average of the k-th column in the PDD matrix, as formalized in Equation 1 [18]:

$$AMD_k(PDD_S) = \frac{1}{m} \sum_{i=1}^{m} d_{ik} \tag{1}$$

Table 1 shows the first 10 AMD values of jobs 00001 and 00026 of the T2 crystal dataset. One can observe that they only differ slightly, often being equal when only considering the first decimal place. When two crystals are equivalent, the AMD values will be identical. However, as discussed earlier, when two crystal structures have the same AMD values, one cannot assume the structures are equivalent due to the incomplete nature of the invariant. In the case of this project, the AMD values are precise to 6 decimal places in order to be able to differentiate between the values in an exact manner.

Table 1: The first 10 average minimum distances of job 00001 and 00026 of the T2 crystal dataset.

Job	AMD_1	AMD_2	AMD_3	AMD_4	AMD_5	AMD_6	AMD_7	AMD_8	AMD_9	AMD_{10}
00001	1.159	1.646	1.681	2.220	2.363	2.497	2.552	2.650	2.734	2.842
00026	1.159	1.641	1.697	2.251	2.368	2.487	2.560	2.676	2.771	2.894

This geometric invariant also requires a set of multiple values in order to be expressive enough. In the case of this project, the first 1000 AMDs are considered, although not all of them may prove to provide a considerable information gain. The set of AMDs is an in-depth abstraction of a crystal's geometric structure and shows great potential for machine learning applications to predict crystal energy. The computation of a set of 1000 AMDs is expected to take only a few milliseconds per crystal instance, significantly less computationally expensive than density functions [18].

1.4 Crystal Energy: An Indicator for thermodynamic Stability

The sets of DFs and AMDs of a crystal, introduced in the previous sections, are used as input to predict the crystal (lattice) energy, which "equals the heat of dissociation of one mole solid into its structurual components" [21]. In other words, when ions are combined into one mole of a compound that forms the crystal structure, the crystal energy is released during this process, essentially indicating the cohesive forces that bind the ions. Crystal energy is measured in $kJmol^{-1}$ and is expected to hold a negative value, as this energy is lost in the crystallization process. In the case of this project, the dataset of crystal structures has a crystal energy range from -223.695 to -123.666 $kJmol^{-1}$. The crystal energy is calculated by "combining the net electrostatic attraction and the Born repulsion energies and finding the internuclear seperation" [14]. Therefore, this calculation is only possible with adequate knowledge of the crystal structure and is quite computationally complex and time-consuming [21].
Although the scientific community commonly refers to this measure as lattice energy, there have been discussions about whether this is the correct use of the term "lattice", which is introduced in Section 1.2 of this paper. Due to a lattice being a mathematical abstraction of a structure, it cannot possibly have an energy that binds the ions within it [15]. Ions are combined into a structure, rather than a lattice, and the use of the term "lattice energy" is self-contradictory [15]. Therefore, within the scope of this paper, this structural energy will be referred to as crystal energy.
During CSP, the crystal energies of all predicted arrangements of a compound are predicted or calculated, generating the so-called crystal energy landscape [3]. The energy landscape is most often represented as a graph with the packing density on its x-axis and the crystal energy on its y-axis, as shown in Figures 6 and 7. The minima of this landscape often indicate the thermodynamically feasible, and therefore stable structures, as this indicates a greater required input of energy to convert

the crystal into gaseous ions [14]. In addition, there may be a few instances that have a crystal energy very close to that of the stable structures, suggesting potential polymorphism. This range of energy between polymorphs is commonly 4 $kJmol^{-1}$ and most often less than 1 $kJmol^{-1}$ [8].

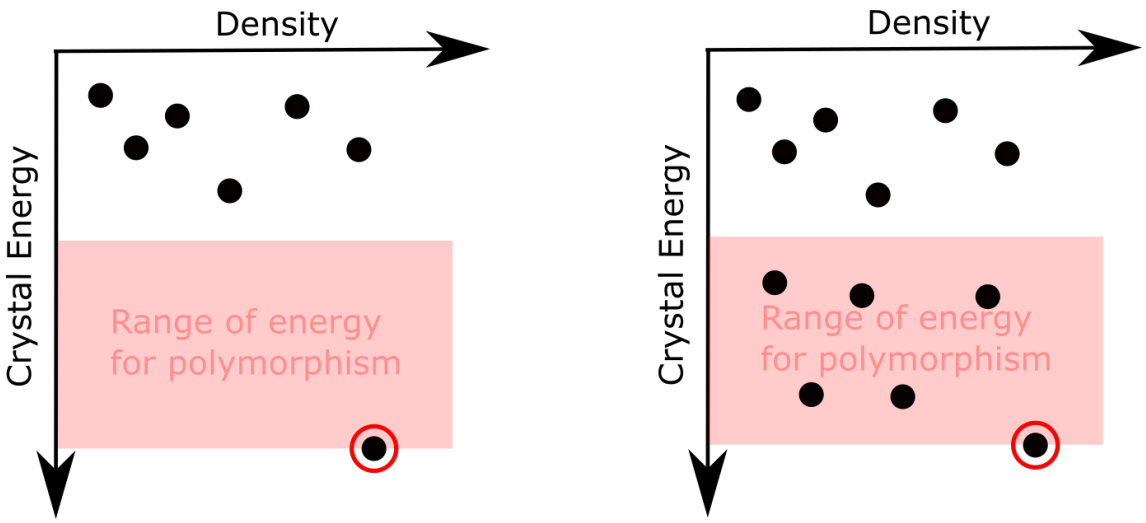


Figure 6: Two examples of a crystal energy landscape (not to scale). *Left:* No indication of polymorphism. *Right:* Indication of polymorphism. [Visualized with Inkscape]

As shown in Figure 6, the left graph shows an energy landscape with a clear global minima, circled in red. This suggests that the minima is the most thermodynamically stable structure and most likely monomorphous, as there is no other arrangement within the range for polymorphism. The other instances are most likely infeasible predictions that cannot be synthesized and can therefore be discarded, decreasing the number of viable candidates. The graph on the right, however, shows the same minima but has multiple structures within the energy range for polymorphism, suggesting a greater probability for a polymorphous substance. In this case, the minima and its potential polymorphs can be kept as candidates for synthesis.
The importance of an accurate method of predicting crystal energy is undeniable, as the calculation of it is extremely computationally expensive and attempting to synthesize all the potential structures is too time consuming and costly, defeating the purpose of crystal structure prediction altogether. Many methods to this problem have been successful, and although still inefficient, they are able to predict this crystal energy with close to sub-$kJmol^{-1}$ accuracy on larger crystal structures, relevant in the production of pharmaceuticals [8].
Shown in Figure 7 is the energy landscape of the T2 dataset, which this project focuses on. After the first step of CSP, the simulated structures have been obtained and the energy ranking process needs to be applied in order to find the most stable structures. The focus of this project is the prediction of this landscape, specifically the crystal energy, for the T2 dataset. The minima of the landscape is clearly visible at a crystal energy of -223.695 $kJmol^{-1}$, representing a stable variant of the crystallized T2 molecule. There are a few additional local minimas in the landscape, also representing proven synthesizable structures.

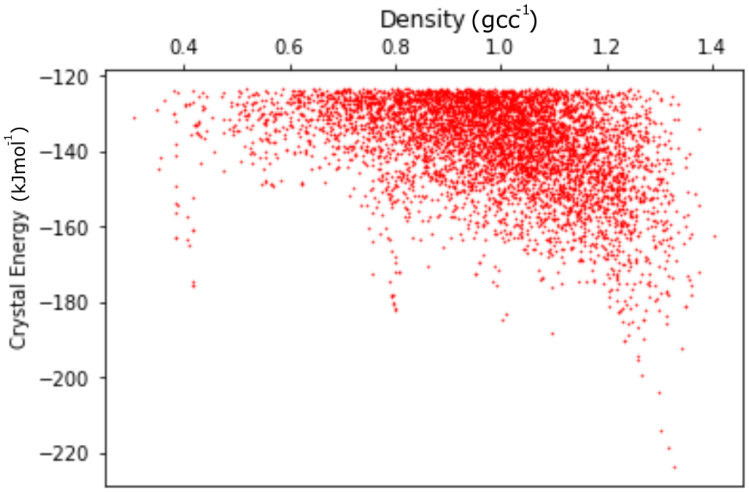


Figure 7: The Energy Landscape of the T2 dataset. [Visualized with Matplotlib]

1.5 Crystal Structure Prediction: A two-step Approach

CSP predicts new crystals by starting with random arrangements of atoms or molecules within a random unit cell and optimizing this by minimizing a complicated crystal energy function, a costly computational process that can take weeks on supercomputers [16, 22]. CSP is broken down into two steps: (1) the random sampling of the crystallographic space, by trial and error, in order to compute feasible crystal structures for the given chemical composition and (2) the ranking of the stability of these predicted structures through the computation of crystal energy [13]. Currently, the two main-approaches for the first step include predictions based on existing knowledge, such as crystal structure databases, and the usage of powerful exploratory algorithms, making predictions with little or no pre-existing knowledge [23]. While these two methods are able to provide a selection of potential crystal structures for the given compounds, they tend to severely over-predict the stable structures [24]. Therefore, they additionally require a method for determining how thermodynamically stable and, therefore, experimentally observable these structures are. This is considered the second step of the problem.
As stable crystal structures are often characterized by low crystal energy, an accurate prediction of this, down to a fraction of a $kJmol^{-1}$, is significant as it can filter the large set of predictions for the limited number of potentially stable structures [25]. This ultimately saves time and money in the lab by reducing the focus to only a few viable candidates [25]. Most recently, levels of accuracy close to this have been achieved with computationally expensive methods, using a hierarchy of calculations that take hours for a single crystal structure [8].
The focus of this project is the prediction of crystal energy (as described in Step 2 of CSP), solely based on the geometric structure, rather than chemical composition and physical properties, of a theoretical crystal structure. It will explore various machine learning methods for this prediction problem and use isometry invariants, acting as "geometric fingerprints" of the actual crystal structures, as the data input.

The ultimate aim is the simplification and acceleration of CSP.

1.6 Previous and related Work

As this project attempts to build on previous scientific work, it is important for existing research on this topic to be discussed. Most notably, this project uses two newly established isometry invariants for periodic point sets as its primary datasets. As introduced in Sections 1.3.1 and 1.3.2, the papers on DFs and AMDs of periodic point sets lay the foundation for the input to this prediction problem [18, 19]. The primary focus of this project is to explore the feasibility of utilizing these geometric invariants for crystal energy prediction, ultimately hoping to bypass previous complex appproaches of obtaining crystal energy. As DFs and AMDs of crystal structures have been proven to be stable isometry invariants, they provide detailed geometric fingerprints of the crystal structures, which can be applied for machine learning prediction purposes.
However, in order to make the assumption that the geometric structure can even provide information that correlates with the crystal energy, one must also consider a proven relationship that connects the two . The most common textbook equation for finding the crystal energy U is the Born-Lande equation, defined as Equation 2 [26]. This equation does not only show the impact of the lattice on the crystal energy, but describes a single ion being affected by the potential fields of all other ions in the lattice [26]. As it is commonly known that the effect of a potential field on a single point is inversely proportional to its distance from the charge, one can conclude that the distance between ions and their overall structure has an impact on the crystal energy [26]. This relationship justifies the hypothesis that a geometric invariant of a crystal structure can be used for prediction of crystal energy, at least to some degree of accuracy, and is worth investigating.

$$U = -\frac{NMe^2}{r}(1 - \frac{1}{n}) \tag{2}$$

, where e is the charge of the electron, r is the shortest distance between oppositely charged ions, M is the Madelung constant, N is Avogrado's number, and n is the Born repulsion exponent [26].

Furthermore, while there have been no attempts to predict crystal energy solely using geometric invariants, past successful solutions to this problem should be discussed. For example, a recent article published in the Journal of Chemical Science measured the similarity between crystal structures to predict the force field relative crystal energy with a sub-$kJmol^{-1}$ accuracy [9]. The authors made use of the training data as reference crystal structures with known crystal energy and compared the data instances for similarity to these references by using a SOAP-REMatch kernel with Gaussian process regression to predict the relative energy [9]. This paper has many similarities to the current project, as it focuses on geometric structure as input and applies regression algorithms to predict the crystal energy. The advantage of using

Gaussian process regression is its effectiveness on small datasets, as well as its output of uncertainty measurements on its predictions. The use of the Gaussian process regression with DFs and AMDs is considered as a viable candidate for this problem. However, the SOAP-REMatch kernel is not considered as this project uses isometry invariants, abstractions of the geometric crystal structure. The SOAP-REMatch kernel is designed for applications on similarity measures between true geometric crystal structures [27]. Instead, other kernels are experimented with.
Similarly and more recently, further attempts of predicting the crystal energy using Gaussian processes have been made [28]. The paper "Multi-fidelity Statistical Machine Learning for Molecular Crystal Structure Prediction" applies an autoregressive Gaussian process to approximate computationally expensive density functional theory calculations, rather than the absolute crystal energy, at around 4.2-6.8% of the original cost. The proposed process consists of a multi-fidelity model that is is recursive and considers correlations between crystal energies that have been calculated using different methods. This model achieves errors of less than 1 $kJmol^{-1}$ using "atom-centered symmetry functions to describe the local (radial and angular) environment of each atom" as input. Although this project focuses on the use of isometry invariants, the promising results from *ref.* [28] further indicate the potential advantages of the Gaussian process in crystal energy prediction, stating that "crystal structure prediction is a well-suited application for a statistical modelling approach".
In addition, the recent DFT+MBD approach has shown excellent results for crystal energy prediction, with a mean absolute error of 2 $kJmol^{-1}$ on larger pharmaceutically relevant crystal structures[8]. However, this approach uses complex hierarchial calculations on molecules and considered many other effects to reach this level of accuracy, taking hours on a single prediction [8]. The article mentions the commonly neglected effects of many-body dispersion, exact exchange, and vibrational free energies in crystal energy prediction having a significant impact on the accuracy [8]. These effects are considered by the DFT+MBD approach, ranking as the most successful first-principles energy ranking strategy in the sixth Cambridge Crystallographic Data Centre blind test [13, 10]. As the approach, suggested in this project, will also be neglecting these effects, it can be hypothesized that a sub-$kJmol^{-1}$ accuracy will be improbable when solely considering the geometric structure. Instead, the goal of this project is a proof of concept for crystal energy prediction using geometric invariants, potentially providing a more efficient method. The DFT+MBD approach is referred to as the current state-of-the-art energy ranking method, used as a reference benchmark measure when evaluating the results of this project and considered when proposing improvement strategies.
Finally, the author wants to consider common as well as modern state-of-the-art techniques in machine learning to address this problem. Due to the nature of the task being a nonlinear regression problem, the use of deep residual neural networks has proven to be a potential option for approaching it [29]. Inspired by ResNet, a popular network architecture in computer vision, the paper introduces its application in nonlinear regression, exchanging convolutional layers with dense layers [29]. This architecture is special in the sense that it is constructed of identity and dense blocks, containing residual connections. These residual connections can be thought of as shortcuts, skipping some connections and transformations and being reintroduced in later layers. Overall, the residual neural network showed significant

improvements in nonlinear regression when compared to common algorithms such as decision tree regression, support vector regression, or artificial neural networks. However, the downside is that this architecture has an extremely large set of trainable parameters and, therefore, also requires large training sets. The network made use of 6,750,000 samples in the training process in order to achieve the high level of accuracy [29]. Therefore, this architecture is not a viable candidate for this project as it is limited by a dataset of only 5679 data instances. Nonetheless, this architecture shows potential for future steps beyond this project, given that more data is collected and made available in order to train more powerful neural networks. This work is elaborated on in Section 7.2.

1.7 Aims and Objectives

In order to formalize the encompassing goals of this research project, some general aims and objectives are composed with the purpose of remaining focused on one primary goal and introducing a set of criteria, by which the success of the project can be measured. While aims, annotated with the letter "A", act as subjective goals, underlining the general intention of the project, objectives, annotated with he letter "O", will be quantifiable and, therefore, objective criteria to the project. The aims and objectives are as follows:

- **A1**: To deliver a proof of concept that crystal energy can be predicted, to some degree of accuracy, based on only the geometrical structure, represented by a geometric invariant.
- **A2**: To explore various machine learning approaches to predicting crystal energy based on a geometric invariant.
- **A3**: To provide a detailed analysis and comparison of the effectiveness of both geometric invariants.
- **A4**: To present the research in an appropriate format, comprehensible to professionals with the relevant scientific background (Data Science, Computer Science, Chemistry).
- **A5**: To develop a general understanding of the topic of crystallography, specifically Crystal Structure Prediction.
- **A6**: To formalize a procedure for predicting crystal energy, based on a geometric invariant.

- **O1**: To investigate which settings for the computation of each geometric invariant serves as optimal input to the prediction problem.
- **O2**: To design a machine-learning algorithm, capable of predicting crystal energy with a mean absolute error of less than 4 $kJmol^{-}1$.
- **O3**: To design a machine-learning algorithm, capable of accurately predicting crystal energy with a training time of less than an hour for the T2 dataset.

- **O4**: To design a machine-learning algorithm, capable of accurately predicting crystal energy with a prediction time of less than 1 Minute per instance.

Regarding Objective 2, the choice of the target accuracy requires an explanation and includes multiple points that have been considered. First, an optimal prediction is in the sub-$kJmol^{-1}$ error range. This represents the best-case results for a prediction algorithm. Second, as mentioned in Section 1.6, the methods in this project ignore the effects of many-body dispersion, exact exchange, and vibrational free energies. Therefore, it is unlikely that the prediction with geometric invariants performs better than the current state-of-the-art methods (which achieve a mean absolute error of 2 $kJmol^{-1}$), making the optimal target accuracy (of sub-$kJmol^{-1}$ error) unlikely [8]. Third and finally, Section 1.4 mentions the crystal energy range of -223.695 to -123.666 $kJmol^{-1}$ in the dataset. This means that an untrained regressor that predicts random values within that energy range reaches the sub-$kJmol^{-1}$ accuracy with 0.02 probability. After close consideration of the three points, an error of 4 $kJmol^{-1}$ is a realistic, yet challenging goal for a trained (non-random) regressor using isometry invariants of crystal structures.

2 Professional Considerations

2.1 The Dataset: T2 by the Cambridge Crystallographic Data Centre

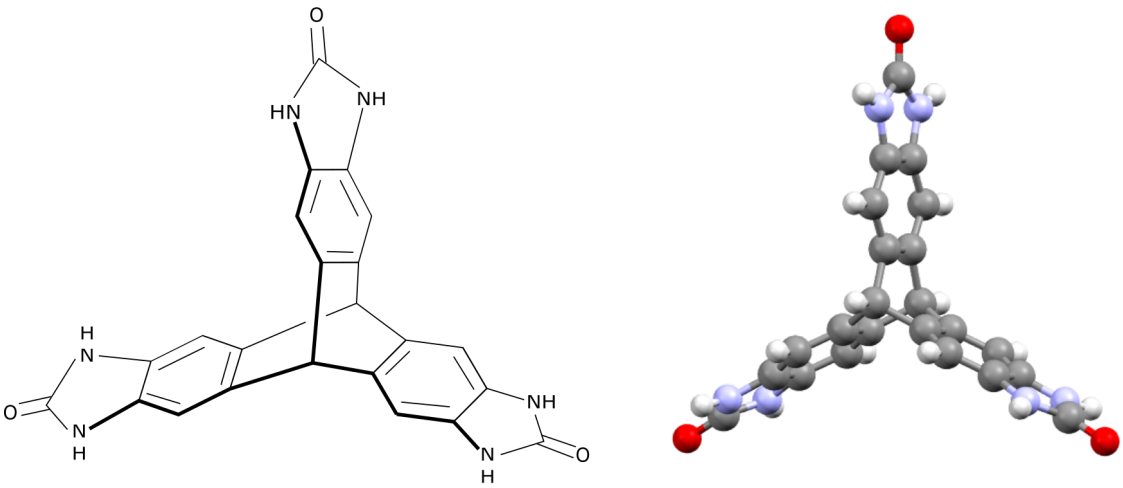


Figure 8: The T2 Molecule (Benzimidazolone Triptycene) [22]. *Left:* A 2D depiction of the molecule. *Right:* A 3D representation of the molecule. The red, white, purple, and black balls represent oxygen, hydrogen, nitrogen, and carbon atoms, respectively. [Visualized with Inkscape (left) and Mercury (right)]

The primary dataset that this paper focuses on is the T2 dataset, as introduced in *ref.* [22]. It consists of 5679 Crystallographic Information Files, simple text files storing the information on the crystals for the T2 molecule [30]. The T2 molecule,

more commonly known as Benzimidazolone Triptycene, is shown in Figure 8 and has the compound formula $C_{23}H_{14}N_6O_3$ [22]. The crystal energy for each prediction has been calculated and ranges from -223.695 to -123.666 $kJmol^{-1}$.
This dataset holds 5679 experimental crystals, of which 5 were actually synthesized and are known as T2, T2-γ, T2-α, T2-β, and T2-δ [22]. These stable structures are found in the spikes of the energy landscape, shown in Figure 6. It is the set of the predicted crystals for the T2 molecule. As this paper focuses on the crystal energy prediction, this dataset is an optimal starting point for approaching this task as it provides crystal structures, each labelled with their corresponding crystal energy. Based on this data, it is possible to train a dataset on the predicted structures and their corresponding energies to predict the crystal energy of unknown structures. The author has full permission to use this dataset for research purposes and the DFs and AMDs are calculated based on the included crystallographic information.

2.2 Dataset of Density Functions of Crystal Structures

One of the main invariant datasets, computed based on the T2 dataset, is the DFs for the 5679 crystal structures. In the case of this project, the first 8 DFs are used, providing a detailed abstraction of the geometric structures that are considered. To make these computations, a C++ program is provided by Philip Smith, one of the authors of the original paper on DFs [19, 31]. This program utilizes Voronoi domains and Brillouin zones to generate a vector output for each DF [19]. In this project, two configurations for the computation are used, ultimately resulting in two DF datasets titled "T2L-C" and "T2L-CO". After computation, there are 5679 indivdual CSV files, consisting of the 8 DFs along their columns and the increasing radii of the spheres along the rows. These files are imported into Python for preprocessing. Full permission to use Philip Smith's code to compute the DFs is given [31].

2.2.1 Variant 1: T2L-C

The "T2L-C" variation is the first configuration of the DF dataset. It is based on the Crystallographic Information Files from the T2 dataset and calculates the first 8 DFs of the molecular centres at a sample rate of 10, meaning that each step in the radii equals an increment of 0.1 $\mathring{A}$. The unique aspect of this configuration is the fact that it calculates the DFs for the geometric centre of the crystal structure, excluding the 3 oxygen atoms on each arm of the molecule, as shown in Figure 8 of Section 2.1. This gives the variant a total of 2104 features after preprocessing, due to the high sample rate. It is hypothesized that the lack of consideration for the three oxygen atoms may make the dataset less expressive, not abstracting the true geometric structure. However, the high sample rate can potentially prove to be extremely beneficial in crystal energy prediction.

2.2.2 Variant 2: T2L-CO

The "T2L-CO" variation is the second configuration of the DF dataset. It is also based on the Crystallographic Information Files from the T2 dataset and calculates

the first 8 DFs of the molecular centres at a sample rate of 5, rather than 10, meaning that each step in the radii equals an increment of 0.2 $\mathring{A}$. The unique aspect of this configuration is the fact that it calculates the DFs for the geometric centre including the first three oxygen atoms of the crystal structure. This means that the oxygen atoms on the end of each arm of the T2 molecule, as shown in Figure 8 of Section 2.1, are included. This gives the variant a total of 936 features after preprocessing, due to the lower sample rate. Although the sample rate is decreased, making the computation less expensive, additional atoms are considered, countering this saving in computational cost. This variant provides insight on the effects of sample rate and the factoring in of the oxygen atoms. It is hypothesized that the consideration of the three oxygen atoms, and therefore the entire geometric structure, makes the data more expressive despite its lower sample rate.

2.3 Dataset of Average Minimum Distances of Crystal Structures

The AMDs of a crystal structure, as introduced in Section 1.3.2, are used as the second isometry invariant in this project. The computed AMDs are also based on the T2 dataset, made up of the 5679 simulated crystals. In the case of this project, the first 1000 AMD values are considered for each crystal structure. Unlike for the DFs, no code was run to compute this data, but rather pre-computed datasets were provided by Mosca and Daniel Widdowson, two of the authors of the original paper on AMDs, who wrote C++ and Python programs to generate the AMDs of point sets and achieved a complexity of near linear time [18, 32, 33, 34]. The data consists of a single CSV file, which includes a row for each of the 5679 crystal instances and a column for each AMD value, ranging from AMD_1 to AMD_{1000}. This file is imported into Python for preprocessing. Full permission to use the computed datasets for research purposes is given by Marco Mosca and Daniel Widdowson . Similarly to DFs, there are two variants of the AMDs that are computed slightly differently; T2L and T2L-CON.

2.3.1 Variant 1: T2L

The "T2L" variant of the AMDs is computed by considering all the atoms within the crystal structures. This means that when representing the crystal structure as a point set, the atomic centres of carbon, hydrogen, nitrogen, and oxygen atoms are regarded to be a point. Therefore, this variant is the true representation of the geometric structure of T2 crystals. The T2L variant consists of the first 1000 AMD values for the 5679 crystal instances.

2.3.2 Variant 2: T2L-CON

The "T2L-CON" variant of the AMD dataset varies with respect to the "T2L" variant by ignoring an atom in the crystal structure during the computation. As hydrogen atoms are quite small in size in comparison to oxygen, carbon, or nitrogen atoms, these have a much smaller impact on the crystal energy. Therefore, the idea

of this variant is to ignore the hydrogen atoms in the structure when representing it as a periodic point set. This means that only the atomic centres of carbon, nitrogen, and oxygen atoms are considered to be points. These two invariants will lead to an insightful comparison on the computation of AMDs and their deployment in crystal energy prediction.

2.4 Third-Party Software Use

In order to efficiently complete this project, several software and libraries are used in order to write the dissertation, view the crystallographic data, and implement the prediction methods. The software Mercury[2] by the Cambridge Crystallographic Data Centre (CCDC) is used to view Crystallographic Information Files and visualize this in 3D [35]. The images of crystal structures are obtained from Mercury. The CCDC states that this system may be used for scientific research as long as it is cited, as stated on the website. Finally, all other images are created with the free open-source vector graphics editor, Inkscape[3], which has a Public License, permitting its free use for any purpose[36].
The primary programming for this project is completed using Python 3.7[4], along with several libraries. All deep learning applications are programmed using the Keras API[5], interfacing with the TensorFlow[6] library [37, 38]. All hyperparameter optimization for neural networks uses the Hyperopt[7] library, which permitted its use in research [39]. In addition, all general machine learning applications use the SciKit-Learn[8] library [40]. The CSV files are loaded and viewed using the Pandas library[9] [41, 42]. Finally, all multi-dimensional array and matrix implementations are enabled by the Numpy[10] library, while all graphing in Python uses the Matplotlib[11] library [43, 44].

2.5 Ethical Considerations

As academic integrity is a key aspect of research, the ethical implications of this project have been closely reflected upon. As the scope of this project does not include any human participants or their personal information, the ethical implications of using individuals' data requires no further consideration. While the topic of crystal structure prediction can have a significant impact on humans, this effect is not direct and introduces no ethical dilemmas. However, the following key points were regarded in the ethical evaluation:

[2] www.ccdc.cam.ac.uk/Community/csd-community/freemercury/
[3] www.inkscape.org/
[4] www.python.org/
[5] Available at: https://keras.io/
[6] www.tensorflow.org/
[7] www.github.com/hyperopt/hyperopt
[8] www.scikit-learn.org/stable/
[9] www.pandas.pydata.org/
[10] www.numpy.org/
[11] www.matplotlib.org/

1. The honest presentation and evaluation of data.
2. The accreditation of external sources used in this paper.
3. The rightful use of third-party software and data.
4. The abidance by the University of Liverpool Policy on Research Ethics[12].

In order to follow ethical consideration 1, the author of this paper presents all data, resulting from the research, truthfully, not making any modifications in order to prove the hypothesis or goal of the project correctly. During the evaluation of this data, the author does not attempt to mislead the reader by coming to false conclusions and making claims that the data does not support. Regarding ethical consideration 2, the author accredits any external sources used in this project. This includes literature, websites, videos, audio, software, and data. These sources are marked accordingly in the text and cited in the bibliography, or pronounced with a footnote. All figures include an indication of the software that was used for their creation. The rightful use of third-party software and data, as mentioned in ethical consideration 3, is extremely important in the field of computer science. As discussed in Section 2.3, the author has the right to use all software resources for research and has cited those that have requested a citation with the corresponding paper. The URLs to all resources have been linked in the footnotes and the software and libraries can be downloaded at no cost. In addition, as discussed in Sections 2.2 and 2.3, the author has permission to use all datasets or code for the computation of these datasets. Finally, ethical consideration 4 includes the "Policy on Research Ethics" of the University of Liverpool. The author has read this, closely considered all its guidelines and implications, and follows this policy throughout the entirety of the project.

3 Regression Problems with Machine Learning

3.1 The Overview

The ability to predict a continuous numeric value, such as the crystal energy of a crystal structure, based on a set of features is considered a regression problem. This form of prediction along with classification make up the two core research branches in the field of machine learning [45]. When approaching this problem with supervised learning, the main tasks include training the model based on a training dataset with known output, adjusting the model to most effectively predict that known output, and feeding the model a test dataset to evaluate how well it can predict this continuous numerical value on new data.
This problem is summarized by Figure 9, which shows some data points that each have a feature x and the continuous numeric value y, which will be predicted. A linear equation can be represented by $y = mx + b$, which transforms the given input feature x into a predicted output y using two parameters, m and b, changing the slope and x-intersect of the line, respectively. Before training, as shown on the left,

[12] www.liverpool.ac.uk/research-integrity/research-ethics/

the function has an extremely high error as it incorrectly represents the given data. During training, the model adjusts the parameters m and b to reduce this error and fit the function to the data points. After training, as shown on the right, optimal values for m and b have been found in order to predict y with minimal error. The equation now fits the data well and can be used for further predictions based on feature x. It is important to note that this example is a simplified example and in the case of this project, x is a vector of features and the function that is optimized to fit this data is not linear.

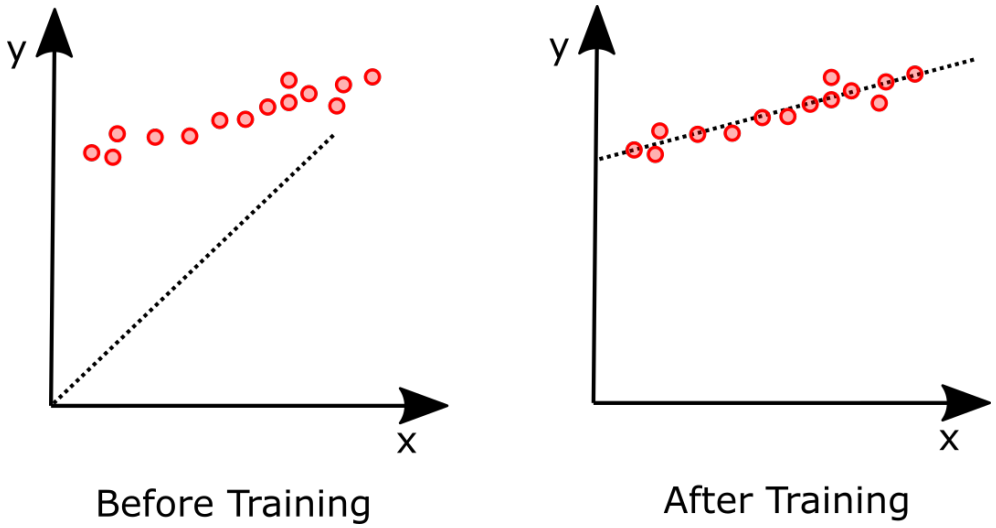


Figure 9: Regression Problem: Fitting a linear function to the data. *Left:* The function parameters have not been adjusted to fit the training data. *Right:* The function parameters are optimal as they have been fit to the training data. [Visualized with Inkscape]

When selecting the data, more specifically the features, for a regression problem, it is of high importance that there is a correlation between it and the value it will predict. As discussed previously in Section 1.6, it is shown that there is an expected correlation between the DFs or AMDs of a crystal structure and its crystal energy. However, it is not known how strong this correlation is and what accuracy can be achieved when predicting the crystal energy with this type of input. In addition, unlike the mentioned example, the function this project attempts to predict is rather non-linear. Therefore, this prediction task can be formalized as a supervised non-linear regression problem using DFs or AMDs as input and predicting the continuous numerical value, the crystal energy, as the output.

3.2 Common Metrics

In order to understand how a model is fit to a dataset, one must also understand the various measures of accuracy or measure associated with regression problems. Unlike in classification, where a prediction is either right or wrong, for continuous numerical values this measure of accuracy would almost always be extremely low and provide no valuable information. Instead, it is better to measure to what degree

or by how much the prediction is wrong, providing a much better evaluation of the model. The most basic measure is the absolute error, which is simply the absolute value of the difference between the predicted output (y_{pred}) and the true output (y_{true}). This measures the magnitude of error for numeric predictions and is shown in Equation 3:

$$AbsoluteError = |y_{true} - y_{pred}| \tag{3}$$

In order to apply the absolute error to an entire dataset, the Mean Absolute Error (MAE) can be used. This measure is the arithmetic mean of the absolute errors on the entire test dataset, consisting of n instances. The lower the measure, the better the model is performing on the dataset. It is represented by Equation 4:

$$MAE = \frac{\sum_{i=1}^{n} |y_{true,i} - y_{pred,i}|}{n} \tag{4}$$

One of the problems with the MAE is that it provides no sense of relativity. For example, when considering two regression problems, one with $y_{true} = 1000$ and one with $y_{true} = 1$, a MAE of 1 can be considered extremely accurate for the first problem but extremely inaccurate for the second problem. The Mean Absolute Percentage Error (MAPE) introduces this relativity to the measure. It is calculated by taking the arithmetic mean of the percentage absolute error relative to y_{true}. It can be formalized as:

$$MAPE = \frac{\sum_{i=1}^{n} \frac{|y_{true,i} - y_{pred,i}|}{y_{true,i}}}{n} \tag{5}$$

Another aspect that may be misleading with the MAE is the equal treatment of all magnitudes of errors. More specifically, the MAE of a model may be relatively low although there are a few predictions with very significant absolute error. In some cases, it may be necessary to use a measure that gives greater weight to larger errors. The Root Mean Squared Error (RMSE) does this by taking the arithmetic mean of the squared differences of y_{pred} and y_{true} and finally taking the square root of that value. This is expressed by Equation 6:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_{true,i} - y_{pred,i})^2}{n}} \tag{6}$$

In the application of crystal energy prediction, this measure can be extremely useful for the evaluation of an algorithm's prediction performance as it is often better to have a model that can generally reach a certain degree of accuracy for all test instances, rather than predict a few instances extremely accurately while predicting other instances with an extremely low accuracy. In the case of this project, various loss functions are considered for training purposes. The loss functions used in the final optimized architectures are introduced in Section 5. For the evaluation, however, all three metrics (MAE, MAPE, and RMSE) are considered as they provide useful insights for the trained model. As the goal is to achieve a consistent regressor, the RMSE score is prioritized.

3.3 Dense Neural Network Regression

One of the machine learning algorithms used in this project is the dense neural network. The elementary building block of a dense neural network is the neuron, often modelled by the perceptron. The concept of the perceptron originates from the field of neuroscience and was intended as a mathematical model for the nervous system, specifically a single neuron, illustrating "some of the fundamental properties of intelligent systems" [46]. It is inspired by the biological neuron, receiving electrical signals from other neurons and firing an output signal when the strength of the input exceeds a certain threshold [46]. These electronic signals are modelled as numerical values. The perceptron can be described as a simple switch, taking a series of inputs and transforming this into a signal response. While this may not be able to express complex information, the field of computing, based on a similar nature by using bits, has shown that a combination and interconnection of many of these switches is able to successfully complete complex processes. As the goal of machine learning is to enable computers to learn, similarly to humans, the perceptron model became an ideal choice. As shown in Figure 10, the model takes a series of n input values $(x_1, ..., x_n)$. Each of these features are given a specific weight $(w_1, ..., w_n)$ and there is a single bias term b, an affine transformation for the case of zero-valued inputs [47]. These values can be combined into the activation score using the Σ function given as [47]:

$$Activation\,Score = \sum_{i=1}^{n} x_i w_i \; + \; b \tag{7}$$

This activation score is then usually fed into an activation function, transforming the continuous number value into either a binary output or normalizing the output within a specific range. Common activation functions include the Sigmoid function, the Binary Step function, the Softmax function, and the Rectified Linear Unit (ReLU) [47]. This model is able to learn simple binary classification problems or linear regressions based on a set of training data. The training data with n features per instance is fed into the perceptron as the input values. The Σ function then calculates the activation score based on the input, its weights, and the bias, and passes

it to the activation function, which produces the final output value. This output value can be compared to the true label of the data instance and, if incorrect, the weights and bias can be adjusted to make an improved prediction next time.

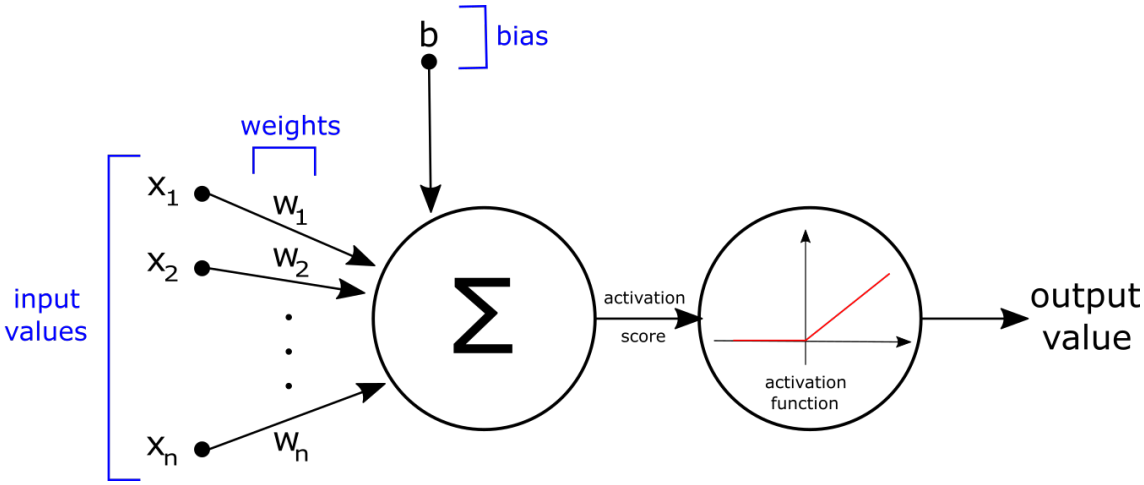


Figure 10: The Perceptron Model: The input features $(x_1, ...x_n)$ are transformed into an output value, using the weights $(w_1, ..., w_n)$, the bias b, and the selected activation function [46]. [Visualized with Inkscape]

This last step emphasizes the importance of a loss/cost function, which measures the magnitude of error between the prediction and the true label, as briefly discussed in Section 3.2. The calculated loss is used in the weight updates by defining the magnitude and direction of the update [47]. In other words, if the prediction is very incorrect then the weight update should be very large and, similarly, if the prediction is only slightly incorrect then the weight update should only be slight. Formally, this is known as gradient-based learning [47]. As the aim of the training process is the minimization of the loss value, this optimization goal can be reached by simply calculating the gradient of the loss function and adjusting the weights to move the loss value closer to the minima [47].
As mentioned, a single perceptron can act as a switch that can be used for binary classification or linear regression. However, when approaching more complex problems such as multi-class classification, computer vision, or non-linear regression, the perceptron often acts as an elementary building block or unit of a neural network. As shown in Figure 11, perceptrons, represented by the blue and red circles, are arranged in interconnected layers, enabling the learning of more complex patterns. While the circles in the input layer simply represent features and their corresponding values for a data instance, the blue circles in the two hidden layers and the single red circle in the output layer are bascially perceptrons. Each arrow pointing at a circle holds a weight and receives an input value from the previous circle. The Σ function is then applied to calculate the activation score, which is activated by an activation function. These values are fed forward through the out-connections to the next layer until a final output is calculated in the output layer. Similarly, the loss or cost of the prediction is then calculated in order to effectively update the weight for each connection.

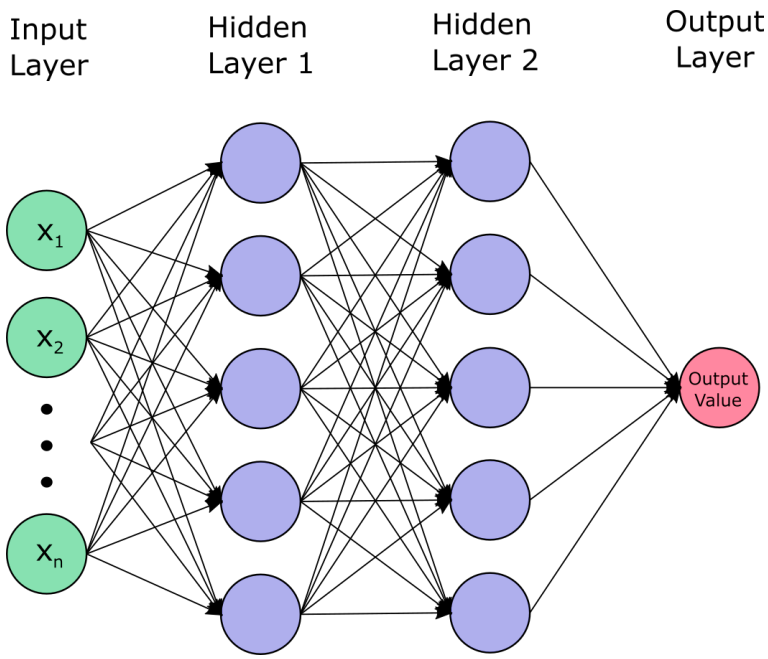


Figure 11: Perceptrons combined into a neural network with an input layer, two hidden layers, and an output layer. [Visualized with Inkscape]

For Neural Networks, the process of the weight update is much more complex due to the high interconnectedness and, therefore, interdependence of the individual layers. The computation of the loss gradient is most effectively completed using an algorithm called Backwards Propagation, which calculates the gradient of each weight relative to the cost function [48]. Back-propagation "allows the information from the cost to then flow backward through the network in order to compute the gradient" [47]. The algorithm recursively applies the chain rule, propagating partial derivatives backwards through the network. This process results in a gradient for each weight in the network, which can then be utilized for weight-updates using algorithms such as stochastic gradient descent or Adam [47, 48, 49].
Most commonly, neural network regression makes use of a dense neural network architecture, in which every neuron of layer k has a weighted connection to every neuron of layer $k+1$. This architecture can learn features from all the combinations of the features of the previous layer. For dense neural network regression in this project, the explored architectures have at least two hidden layers and the layer sizes will vary by experiment. Hyperparameter optimization is used in order to explore the space for an optimal architecture.
Besides the densely connected layers, the use of dropout layers is also considered in the scope of this project, providing an inexpensive, yet powerful method for regularization [47]. This layer randomly multiplies connections between two dense layers by zero, essentially "dropping" a connection during the training process. This regularizes the network by causing each hidden unit to perform well, independent of which other hidden units are included in the model. The dropout layer randomly applies noise, preventing the model's focus on single features, and instead encouraging the focus on a range of features [47]. For example, in the case of this project, there could potentially be a high correlation between the first DF or the first AMD value and the crystal energy. However, for the purpose of regularizing the model, it can be

beneficial to sometimes drop the feature value for the first DF or AMD in order to force the model to learn how to predict the crystal energy based on other features, ultimately preventing a high dependence on a single feature.

3.4 Gaussian Process Regression

Gaussian process regression is another approach to the regression problem, considered in this project. Not only has this algorithm been proven to be successful in crystal energy prediction, as discussed in Section 1.6, but it is generally a popular model for non-linear regression problems [9, 50]. The reason for this is its characteristics of being expressive, interpretable, and avoidance of over-fitting [50].
In simple terms, the Gaussian process is a generalization of the Gaussian probability distribution [51]. It uses kernels to measure the similarity between sampled data, which can then be used to make predictions for new data instances [9]. In essence, a prior probability is given to every function in the initial function space, affected by the choice of kernel that represents a prior belief about the relationship between the data and the output value. This creates a distribution of functions. With the new information of the training data, the posterior probability of each function is calculated using Bayesian inference. The higher this posterior probability, the more likely the function fits the observed data [51]. However, since there is an infinite space of functions that can potentially fit any given dataset, which would be impossible to compute in finite time, the Gaussian process only considers the properties of a finite number of points, which will end up resulting in the same output as if the infinite number of points is taken into account [51]. A mean function is then calculated based on all the weighted (by their posterior) functions, with the most likely functions having the best fit. Gaussian process regression does not only output a scalar value, calculated from the mean of this probability distribution of functions, but also the standard deviation of the prediction, which is smallest near observed data instances. This standard deviation can be thought of as the magnitude of deviation of all the functions relative to the mean function. This means that when more observed datapoints are added, the mean function will pass through them and have a uncertainty of zero at the known points as all the functions pass through those points[51].
Mathematically speaking, the Gaussian Process regression is defined by its mean vector μ and its covariance matrix Σ and is denoted as $\mathcal{N}(\mu, \Sigma)$ [51]. More specifically, μ is the expected value of the distribution of functions and Σ holds information on the variance along each dimension and the correlation between the various features [51]. The covariance matrix is computed using a covariance function, often referred to as a kernel, which sets the shape of the distribution and measure of similarity between data instances [51]. This kernel is applied to all pairs of points and returns a scalar measure of similarity of the two points. Therefore, the value Σ_{ij} of the covariance matrix is the covariance between feature i and j, providing insight into how similar these variables behave [51]. In the case of this project, the Rational Quadratic kernel is primarily used, as implemented in the SciKit Learn library[13]. This kernel is defined as:

[13] www.scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RationalQuadratic.html

$$k(x_i, x_j) = (1 + \frac{d(x_i, x_j)^2}{2al^2})^{-a} \tag{8}$$

, where $d(x_i, x_j)$ is the Euclidean distance between the two points and a and l the alpha and length scale parameters, respectively. However, various other kernels and their sums and products are considered during optimization and are discussed in Section 5. The kernel also defines the prior distribution, which can be seen when no observed data points have been included, as shown in graph a) of Figure 12. Once the model has been fit to the training points, predictions can be made by sampling from the distribution using marginalization of the individual random variables to output the corresponding mean and standard deviation for the specific instance [51]. Due to the lengthwise constraints on this dissertation, this process cannot be explained in more detail as it would require an extensive introduction into Bayesian probability and Multivariate Gaussian Distributions. However, the essence of the Gaussian Process has been touched upon, providing sufficient knowledge for the general understanding of the method. The author does not implement the Gaussian process regression, but rather makes use of the implementation from Sci-Kit Learn[14]. This library implementation is based on Algorithm 2.1 "Gaussian Process for Machine Learning", authored by Rasmussen and Williams [51].

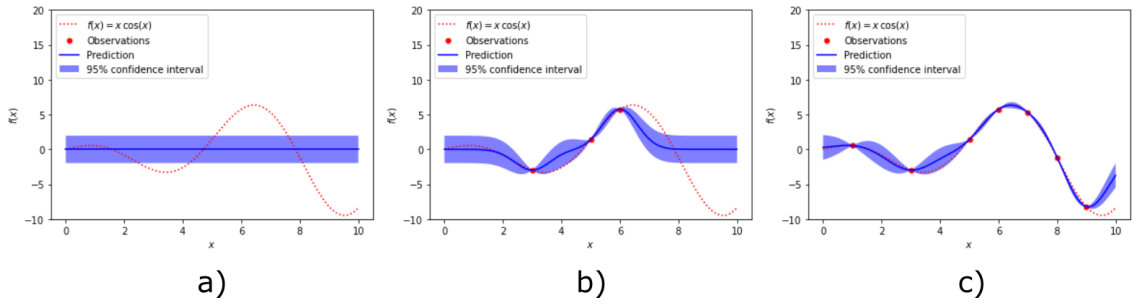


Figure 12: Gaussian Process fitting on varying numbers of observed data points. *Left:* A pure representation of the kernel, as no training data has been introduced. *Center:* Fit on three observed points. *Right:* Fit on 6 points. [Visualized with Matplotlib]

This Gaussian Process of fitting is shown in Figure 12, attempting regression for the function $f(x) = xcos(x)$, which is represented by the red dotted line. In graph a), no observed data points are present, and, therefore, for all features x the mean of the distributions, represented by the blue line, is equal to 0 and has a constant confidence interval, as shown by the shaded area. Initially, with the lack of any training data, the shape of the mean function is defined by the kernel. Graph b) uses 3 observed data points, indicated by the red points and making the mean function pass through those points with a decreased uncertainty in the shape of a Gaussian distribution. Finally, graph c) has a total of 6 observed points, fitting the mean of the distribution to the function very well with low uncertainty for most

[14] www.scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html

input values. This shows how the Gaussian process learns from observations to fit to a specific function. When making a prediction for a given value of x, the mean function value will be returned, along with the given uncertainty for that instance, expressed by a standard deviation of all the considered functions. It is important to note that this example only considers a single feature x and in the case of this project at least a hundred features will be considered, making it much more difficult to visualize in a useful manner.

3.5 Random Forest Regression

The third and final algorithm, considered in this project, is random forest regression. Again, this model can be broken down into an elementary building block called a decision tree. A decision tree is a "hierarchical model composed of discriminant functions, or decision rules, that are applied recursively to partition the feature space of a dataset" [52]. It is a binary tree structure with a single root node, which represents the entire sample set that will be divided into subsets, decision nodes, which split datasets into subsets based on a decision boundary, and terminal or leaf nodes, which return the predicted value [52].

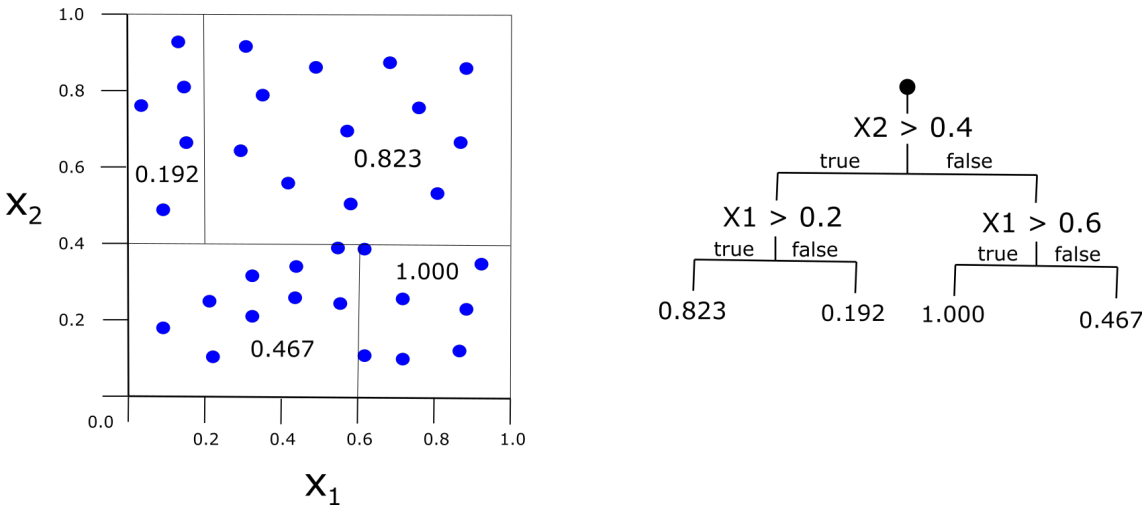


Figure 13: A decision tree for regression, fit on a set of data with two features. *Left:* The decision boundaries, shown on a plot. *Right:* The decision boundaries, represented as a tree structure. [Visualized with Inkscape]

As shown in Figure 13, the graph on the left represents the dataset with two features, one on each axis. This dataset is partitioned into regions, seperated by decision boundaries and defined by their label (the printed numbers in each region). The various combinations of features x_1 and x_2 and their decision boundaries are used to recursively partition the feature space until each partition represents a single label [52]. This process creates the decision tree, shown on the right of Figure 13. Each decision boundary is represented by a decision rule. If, for example, the features of a test instance are $x_1 = 0.9$ and $x_2 = 0.3$ then the first decision rule of the tree would evaluate as false and the second decision rule as true, ultimately

returning a prediction of 1.000. In summary, the decision tree aims to provide a set of decision rules that partition the feature space in order to robustly predict scalar values in a hierarchical manner [52].
As one can assume from the name, a random forest is an ensemble learning approach, combining multiple decision trees and first introduced by Leo Breiman [53]. Breiman defines the random forest as "a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, ...\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular [prediction] at input x" [53]. Random forests make use of Breiman's bootstrap aggregating predictor method, which includes random sampling of small subsets of the dataset in order to train multiple versions of a predictor and combine these into an aggregated predictor [54]. In addition, not only are subsets of the data used, but also subsets of the features in order to assure that all features and their combinations are considered and no single feature dominates the decision boundaries. The motivation for this is slightly similar to the dropout layer in neural networks. As single decision trees tend to overfit easily and are extremely sensitive to specific data, the ensemble of multiple decision trees trained on varying subsets of the data and features can be quite advantageous [55]. Each of the n decision trees of the random forest vary slightly in the sense that they train on randomly sampled subspaces of the data and feature space, introducing an aspect of randomness that encourages generalization [55]. The predictions of the n trees are then averaged to get the final output of the model. Random Forests are an extremely popular approach to regression problems and commonly provide relatively accurate results in most applications. Within this project, the random forest regressor is implemented using the Sci-Kit Learn library[15].

4 Methodology

4.1 The Data

4.1.1 Preprocessing

General Preprocessing
Various normalization techniques for the feature and label data are considered during the preprocessing step. The two primary techniques used in this project are MinMax and Gaussian scaling [56]. The MinMax scaling uses the maximum and minimum value of a set of data to transform the values into a range of $[0, 1]$ while maintaining the relative relationships between the values [56]. Given the dataset X with its instances x_i, the scaled value x_i' is defined as:

$$x_i' = \frac{x_i - min(X)}{max(X) - min(X)} \tag{9}$$

As many machine learning algorithms, especially neural networks, work optimally with low values, this is a common approach to transforming the dataset into a specific

[15] www.scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

range while maintaining the information within it. Another scaling technique in this project is the Standard scaling, commonly referred to as Gaussian normalization. It transforms the data of each feature to a distribution with a mean of zero and a unit variance, making it easier to compare two features by ignoring their scale [56]. A normalized value x' is defined as:

$$x' = \frac{x - \mu}{\sigma} \tag{10}$$

, where μ is the mean and σ the standard deviation of X. Finally, some data is additionally L2-normalized, meaning that each feature vector is transformed to have a unit Euclidean norm. Depending on the data and the machine learning method, one of the three suggested techniques will be applied in order to optimize the results. The final choice is introduced in the optimized architectures in Section 5.

Density Functions

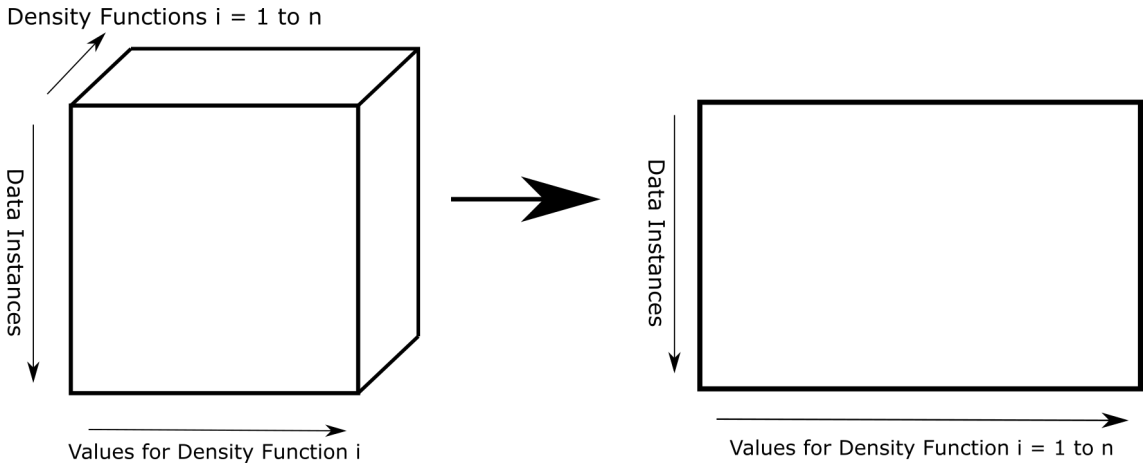


Figure 14: Preprocessing of Density Functions: A transformation from the 3-dimensional tensor to the 2-dimensional matrix, serializing the density ufnctions into a single vector. [Visualized with Inkscape]

Originally, after the computation of the DFs, the feature dataset consists of 5679 CSV files. Each CSV file represents a crystal structure and has a total of 9 columns, the radii label plus the 8 density functions, and a varying amount of rows depending on the choice of radii increments and the domain of the 8^{th} DF (until its value reaches zero again). These CSV files are individually imported and accumulated in a single three-dimensional tensor, defined by its shape $Row \times Column \times Depth$. As the domain for the DFs vary within the dataset, it is important to first evaluate what the largest domain is in order to set a constant domain size for the functions. If each crystal instance had a varying domain size for its functions there would be an inconsistency in the data layout, most likely complicating the training process. The code for the import of the T2L-C variant and the T2L-CO variant is available

on GitHub[16][17]. In the case of the "T2L-CO" dataset, as introduced in Section 2.2.2, the standard domain length is 117 values at an increment of the radii of 0.2 units. This results in an initial three-dimensional tensor of size $5679 \times 117 \times 8$.
As shown in Figure 14, the previously mentioned three-dimensional tensor is depicted on the left, with $n = 8$. The next preprocessing step is the transformation into a two-dimensional matrix by placing the $n = 8$ DFs of length 117 each into series, resulting in a vector of length 936 per instance. Ultimately, the transformation is complete with a new two-dimensional matrix of size 5679×936. The code for this step is available as *Appendix B*. As only a few data instances have data for the full domain of 117 values, there are some missing values in the data. For example, a specific data instance with a domain of 110 values has 7 missing values at the end of each function. These missing values can be filled with the value of zero.
Further data preprocessing steps, such as scaling and normalization, are dependent on the specific machine learning method and are discussed in Section 5, along with the optimal architecture . The labels for the data are imported from a CSV file and kept in the original vector form. The values are matched to the data instances using the job ID associated with each crystal structure. Therefore, a data instance i in the feature matrix has its label at index i of the label vector. Depending on the machine learning method, the labels may be scaled for the training process and inverse scaled for the evaluation in order to improve the accuracy.

Average Minimum Distances

While the label data for the two AMD datasets, as introduced in Section 2.3, undergoes the same process as that of the density functions, the feature data requires much less preprocessing. The data for the entire 5679 instances comes in a single CSV file, which can be imported directly into Python. This import results in a two-dimensional matrix of shape 5679×1000. The only preprocessing step is the selection of the number of AMD values that is used. In the case of this project, the first 100 Average Minimum Distances of each instance are used most often (although all 1000 are considered), resulting in a sliced 5679×100 matrix. However, other feature sizes are experimented with in the optimization process. The final matrix has no missing values and is ready for use. Depending on the machine learning method, scaling and normalization may be applied on the dataset. However, this is explained in Section 5 in more detail.

4.1.2 The Train-Validation-Test Split

An important aspect of any machine learning method is the dataset split into training and test data. Additionally, it is advantageous to have a validation dataset for the accuracy monitoring during training, especially for neural networks. As the dataset of the T2 crystals is relatively small, especially for deep learning methods, the author chose to keep a relatively large portion of the dataset for training. The dataset split is as follows:

[16] github.com/JRopes/CrystalEnergyPrediction/blob/main/Preprocessing/DataImporter.py
[17] github.com/JRopes/CrystalEnergyPrediction/blob/main/Preprocessing/DataImporter_CO.py

- **Training Dataset:** 81% of the dataset; 4600 instances
- **Validation Dataset:** 9% of the dataset; 511 instances
- **Test Dataset:** 10% of the dataset; 568 instances

It is important to note that the dataset is shuffled before the split in order to assure a random distribution of crystal structures and their relative crystal energies. This split guarantees a large training dataset while still providing sufficient evaluation capabilities. In addition, as random forests and Gaussian processes do not make use of a validation dataset, the validation split is simply added to their training data.

4.2 Prototyping and Evaluation Strategy

Like most machine learning applications, the development process includes an iterative prototyping and experimentation approach. In this project, the development process is inspired by the Cross Industry Standard Process for Data Mining (CRISP-DM), which can be extremely efficient for machine learning applications in a new field of study, such as Crystallography [57].

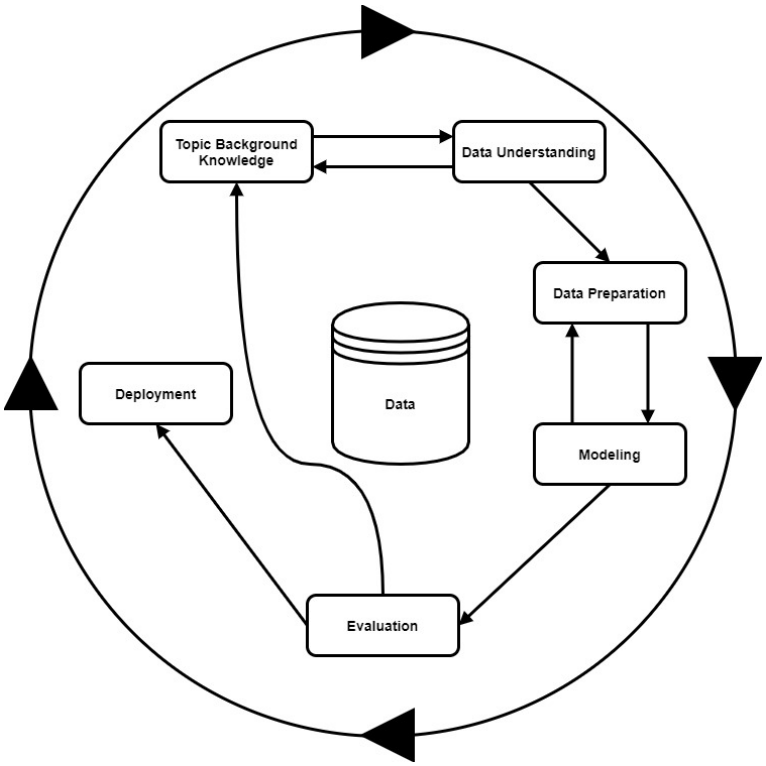


Figure 15: CRISP-DM Development Model, as inspired by *ref.* [57]. [Visualized with Inkscape]

As shown in Figure 15, the development strategy emphasizes the importance of topical background knowledge, its application to the dataset understanding, and the concluding prototyping attempt, which is ultimately evaluated on the test data. The related topics of crystallography and chemistry were studied in order to understand

the scope of the problem. This knowledge was further applied to the data, enabling a deep understanding of its meaning and significance, while allowing new questions to arise, which were answered by further research. Once the background knowledge and understanding of the data was acquired, the efficient preprocessing of the data was possible, already having an adequate understanding of what transformations the data needs to undergo for a machine learning application. In addition, various prototype models, including dense neural networks, Gaussian processes, and random forests were built and experimented with. The observations from the experimentation lead to further adjustments to the data preparation and models in order to optimize the results. Finally, the models were evaluated on the test data and were either sent into another iteration of the process or finalized, as they have met the requirements or goals. This cycle of gaining an understanding of the topic and data along with learning from experimentation enabled the development of an efficient final model that is ready for deployment.
An important aspect of the CRISP-DM model is the evaluation step, which acts as a decision node between an additional iteration of prototyping and the final model deployment. In the case of this project, each model was evaluated on the 568 test data instances, as introduced in Section 4.1.2, using the MAE (Equation 4), MAPE(Equation 5), and RMSE (Equation 6), as defined in Section 3.2. The evaluation process aims to reach minimal values for the mentioned metrics by optimizing data preparation and hyperparameters for the models. This allowed the conclusion on an optimal model and configuration for each invariant, as well as an overall approach for predicting crystal energy.
In order to achieve a statistical significance on the final results, each optimized model underwent multiple runs to calculate a statistical mean with an uncertainty interval. In the case of this project, each model was run a total of 10 times. The accuracy is expected to vary slightly due to the random shuffle of the data, resulting in different distributions of the data, and randomized methods within the model, such as the bootstrapping in random forests or random weight initialization in the neural network. For each metric, the mean was calculated for the 10 runs and the uncertainty was found by calculating the standard deviation over the 10 runs. This provides an accurate measure of each model's general performance.

5 Optimized Architectures

5.1 Random Forest Regression

For the case of predicting crystal energy using a random forest regressor, the SciKit-Learn implementation[18] was utilized and optimized on the prediction problem. The primary parameters that were experimented with include the number of features (number of DFs or AMDs), the number of decision trees within the random forest (between 50 and 600 in steps of 50), the loss function used to build the random forest (Mean Squared Error and Mean Absolute Error), the scaling and normalization of the data, and the size of the sub-sample used for training each tree (0.50 to 1.00 in steps of 0.05). Using grid search, all plausible combinations of these parameters

[18] scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

were experimented with in order to get optimal results on the test data, evaluated using the RMSE and MAE.

Table 2: Optimized Parameters for Random Forest Regression, found through experimentation and grid search.

Input (Invariant)	Data Preprocessing			Random Forest Parameters		
	Feature Size	Scaling	Normalization	Forest Size	Subsample Size	Loss Metric
Density Functions (T2L-C)	2104 (8 Density Functions)	Standard on Features	L2	500	0.95	MSE
Density Functions (T2L-CO)	936 (8 Density functions)	Standard on Features	L2	400	0.95	MSE
Average Minimum Distances (T2L)	100	Standard on Features	None	230	1.0	MSE
Average Minimum Distances (T2L-CON)	100	Standard on Features	None	230	1.0	MSE

Density Functions

As summarized in Table 2, for the T2L-C variant of the DFs the random forest achieved optimal results using all 2104 features, meaning all 8 DFs. This feature data was scaled using the Standard scaler (Equation 10), as introduced in Section 4.1.1, and normalized with the L2 norm. The random forest consisted of a total of 500 trees, each randomly sampling 95% of the training set and using the Mean Squared Error metric for optimization of the decision boundaries. The code for this optimized random forest can be found on GitHub[19].

For the T2L-CO variant, the random forest also achieved optimal results with using all 8 DFs, consisting of 936 features. These features were scaled using the Standard scaler (Equation 10) and normalized on the L2 norm. Due to the smaller feature space, this optimized random forest only required 400 trees. Similarly to the T2L-C variant, each tree randomly sampled 95% of the training set and used the Mean Squared Error metric for optimizing its decision boundaries. The code for this implementation can be found on GitHub[20].

Out of the two variants, the random forest regressor, optimized on the T2L-C dataset, achieved the best results in terms of both RMSE and MAE. It should be acknowledged that the use of DFs in random forests required a much larger number of trees and features than with the use of AMDs, making the overall training process significantly slower. The random forest architecture for DFs, however, used the full bootstrapping capabilities, not only randomly sub-sampling the feature space, but also the training sample space.

[19] github.com/JRopes/CrystalEnergyPrediction/blob/main/DF_T2L_C_RandomForest_Predictor.ipynb

[20] github.com/JRopes/CrystalEnergyPrediction/blob/main/DF_T2L_CO_RandomForest_Predictor.ipynb

Average Minimum Distances

The optimized data preprocessing strategy and parameters for the random forest regression using the the T2L variant of AMDs are summarized in Table 2. The results show that an optimal RMSE value was achieved using the Standard Scaler (Equation 10), as introduced in Section 4.1.1, on the first 100 AMD values. Notably, this scaling was only applied to the features, rather than the labels. The architecture made use of 230 trees, each sampling the entire training set but only a subset of the features for each tree and using the Mean Squared Error metric for optimization. While the MAE was slightly better with 190 trees, the difference is negligible and a low RMSE score is the highest priority in this problem. The implementation for this random forest for the T2L variant can be found on GitHub[21],

For the T2L-CON variant of the AMDs, the random forest achieved optimal results with the exact same parameters as the T2L variant. However, it did not achieve the same accuracy. The implementation of the random forest for the T2L-CON variant can be found on GitHub[22]. Generally, the use of L2 normalization seemed to decrease the accuracy of the regressor. In addition, the results show that the bootstrapping in random forests did not improve the metrics and instead only random subsets of the 100 features are used in order to guarantee varying decision trees and regularization. The Mean Absolute Error loss function did not only significantly slow down the training process, but also performed worse in comparison to the Mean Squared Error metric.

5.2 Gaussian Process Regression

Table 3: Optimized Parameters for Gaussian Process Regression, found through experimentation and the automatic optimization in the Gaussian process algorithm.

Input (Invariant)	Data Preprocessing			Gaussian Process Parameters		
	Feature Size	Scaling	Normalization	Kernel	Alpha	Length Scale
Density Functions (T2L-C)	2104 (8 Density Functions)	Standard on Features and Labels	None	Rational Quadratic + Squared Exponential	1.0	1.0
Density Functions (T2L-CO)	936 (8 Density functions)	Standard on Features and Labels	None	Rational Quadratic + Squared Exponential	1.0	1.0
Average Minimum Distances (T2L)	100	MinMax on Features and Labels	None	Rational Quadratic	1.0	1.0
Average Minimum Distances (T2L-CON)	150	MinMax on Features and Labels	None	Rational Quadratic	1.0	1.0

The optimization process for the Gaussian process regression, as implemented by SciKit-Learn[23], focused on various preprocessing methods and kernel parameters.

[21] github.com/JRopes/CrystalEnergyPrediction/blob/main/AMD_T2L_RandomForest_Predictor.ipynb
[22] github.com/JRopes/CrystalEnergyPrediction/blob/main/AMD_T2L_CON_RandomForest_Predictor.ipynb
[23] scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html

In regards to data preprocessing, the feature size (number of DFs or AMDs), the scaling method, and the use of normalization were the most impactful factors. The kernel choice, as briefly discussed in Section 3.4, as well as its parameters played a vital role in the fitting of the data. The kernels that were considered for this problem are the rational quadratic[24] (Equation 8), the squared exponential[25], the matern[26], the linear[27], and the polynomial kernel, which is simply the product of linear kernels. In addition, sums and products of kernels were also experimented with. The implementation of the Gaussian process regressor automatically finds the optimal kernel parameters during the fitting process. This made the optimization process very simple as the focus is reduced to kernel and preprocessing methods.

Density Functions

The preprocessing techniques and architecture parameters for achieving optimal results with the Gaussian process, using the T2-C variant of the DFs, are summarized in Table 3. In terms of the feature selection, all 8 DFs were used, making the feature vector a size of 2104 elements. The features, as well as the labels, were scaled using the Standard scaler (Equation 10), as introduced in Section 4.1.1. No L2-normalization is applied to the feature data as it significantly reduced the accuracy of the model. Unlike the Gaussian process for AMDs, which used a single kernel, the model for the DFs made use of the sum of two kernels; the Rational Quadratic kernel and the Squared Exponential kernel. The alpha and length scale parameters are both set to 1.0. The implementation of this model can be found on GitHub[28].
For the T2L-CO variant, 936 features were used, this being the equivalent of all 8 DFs. Similarly to the implementation for the T2L-C variant, the features and labels were scaled with the Standard scaler (Equation 10) and the sum of the Rational Quadratic kernel and the Squared Exponential kernel deemed optimal results. The implementation for this model is available on GitHub[29]. Overall, the model for the T2-C variant achieved the best results for DFs and the Gaussian process in terms of RMSE, while the model for the T2L-CO variant performed slightly better in terms of the MAE. It is worth noting that the margins are minimal and that both variants of DFs provided similar results.

Average Minimum Distances

Overall, the Gaussian process with AMDs showed the biggest potential out of all invariants and algorithms. Therefore, in-depth testing was completed on the kernels and feature sizes of the two variants in order to optimize this model further. The detailed results of these optimization experiments can be found in *Appendix A* of this report. This optimization process made use of 80% training data and 20% test data.
As shown in Table 3, optimal results were obtained by using the first 100 AMD values of the T2L variant, scaled in a range of 0 to 1 using the MinMax scaler

[24] scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RationalQuadratic.html
[25] scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html
[26] scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.Matern.html
[27] scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.DotProduct.html
[28] github.com/JRopes/CrystalEnergyPrediction/blob/main/DF_T2L_C_GaussianProcess_Predictor.ipynb
[29] github.com/JRopes/CrystalEnergyPrediction/blob/main/DF_T2L_CO_GaussianProcess_Predictor.ipynb

(Equation 9). In addition, not only were the features scaled for the Gaussian process, but also the labels, which proved to significantly improve the fitting process. For evaluation, however, the predictions were inverse scaled in order to represent the true crystal energy values. Regarding the kernel, the Rational Quadratic function, as introduced in Section 3.4, proved to fit the data best with an alpha parameter of 1.0 and a length scale parameter of 1.0. This implementation is available on GitHub[30]. It is worth mentioning that some of the parameter adjustments resulted in negligible differences and that the most important factors were the feature size and kernel choice.
For the T2L-CON variant, which resulted in a slightly worse accuracy, the optimal model used the first 150 AMD values. Similarly to the model for the T2L variant, the MinMax scaler (Equation 9) was applied on the features and labels and a Rational Quadratic kernel with alpha and length scale parameters of 1.0 was passed to the Gaussian process. The code for this model is available on GitHub[31].

5.3 Dense Neural Network Regression

Table 4: Optimized Parameters for Dense Neural Network Regression, found through experimentation and random search.

Input (Invariant)	Data Preprocessing			Neural Network Parameters	
	Feature Size	Scaling	Normalization	Batch Size	Loss Function
Density Functions (T2L-C)	2104 (8 Density Functions)	Standard on Features	None	128	Huber
Density Functions (T2L-CO)	936 (8 Density functions)	Standard on Features	None	128	Huber
Average Minimum Distances (T2L)	100	MinMax on Features	None	128	Huber
Average Minimum Distances (T2L-CON)	100	MinMax on Features	None	128	Huber

The optimization process for the Dense Neural Network regression made use of the Hyperopt[32] implementation for Keras, which uses random search to select a combination of optimal parameters in order to minimize the loss on the validation data set [39]. With regards to data preprocessing, the optimization process focused on the feature size (number of DFs or AMDs), the scaling technique, and the normalization of the features. The primary focus within the optimization of the actual algorithm included the architecture, such as the number of layers and neurons within them, the loss function, and the optimal batch size for the training process. The number of epochs is very dependent on the weight initialization and data split and, therefore, no optimal value is listed. The popular Adam optimizer was used for all experiments, as it proved to give superior results in all cases.

[30] github.com/JRopes/CrystalEnergyPrediction/blob/main/AMD_T2L_GaussianProcess_Predictor.ipynb
[31] github.com/JRopes/CrystalEnergyPrediction/blob/main/AMD_T2L_CON_GaussianProcess_Predictor.ipynb
[32] github.com/hyperopt/hyperopt

Density Functions

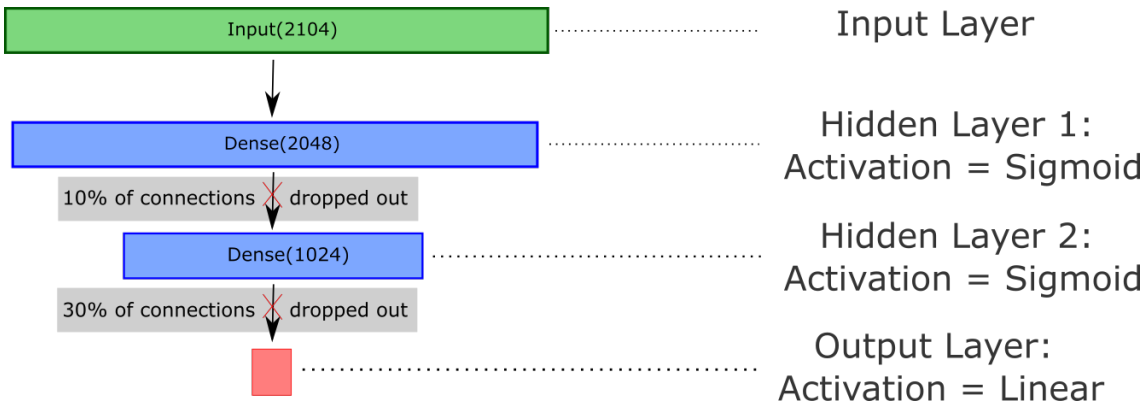


Figure 16: Optimal Dense Neural Network Architecture for Density Functions, specifically the T2L-C variant. [Visualized with Inkscape]

As shown in Table 4, the optimized dense neural network used the first 8 DFs of the T2L-C variant, resulting in a feature vector of size 2104. The features were scaled using the Standard scaler (Equation 10) but not L2-normalized. The labels underwent no preprocessing steps. The optimized architecture is portrayed in Figure 16, with the input layer consisting of 2104 feature inputs. This is followed by the first hidden dense layer, consisting of 2048 neurons and activated by a Sigmoid function. 10% of the connections to the next layer are dropped to avoid overfitting. The second hidden layer has a total of 1024 neurons and is also activated by a Sigmoid function. This is followed by another dropout layer that removes the signal from 30% of the connections. The final output layer is a single densely connected neuron, activated by a linear function and outputting a scalar value. This value represents the predicted crystal energy and does not require inverse scaling. Overall, this architecture has a total of 6,410,241 trainable parameters, mainly due to the large input vector, and is available on GitHub[33]. Along with the Adam optimizer, the piecewise Huber loss function was used in the training process [58]. The Huber loss function is formally defined as:

$$H_\alpha(x) = \begin{cases} \frac{1}{2}x^2, & |x| \leq \alpha \\ \alpha(|x| - \frac{1}{2}\alpha), & |x| > \alpha \end{cases} \quad (11)$$

, where α is a positive real number that works as a threshold between the two loss functions and the variable x is defined as the absolute error (Equation 3), as introduced in Section 3.2, between y_{pred} and y_{true} [58]. In the case of this setup, the variable $\alpha = 1.0$. The Huber loss is a piecewise combination of the commonly used L_1 and L_2 norms and compromises between sensitivity to outliers and the characteristic of being differentiable at any value [58]. This loss function proved to be optimal in the training process.

[33] github.com/JRopes/CrystalEnergyPrediction/blob/main/DF_T2L_C_DNN_Predictor.ipynb

The model for the T2L-CO variant performed slightly worse. The data was pre-processed in the same manner as for the T2L-C variant but due to the difference in feature size the optimal architecture is slightly different. After the input layer, a dense connection is made to the next layer, consisting of 256 neurons and activated by a Sigmoid function. This is followed by a dropout layer, randomly zeroing out 20% of the connections. The second hidden layer consists of 1024 neurons and is again activated by a Sigmoid function. This layer is densely connected to the third hidden layer with 512 neurons and activated by a Sigmoid function. The final output layer is a single neuron with a linear activation function. Overall, this model has a total of 1,028,353 trainable parameters, much smaller than the implementation for the T2L-C variant. The model also makes use of the Huber loss function (Equation 11) and is optimized by the Adam algorithm. The final implementation is available on GitHub[34].
In order to optimize the fitting process, an early stopping call back function was implemented for both models. This has a sensitivity of 20 epochs and stops the training process when the validation loss no longer improves. The learning curves of the training processes showed no signs of over- or underfitting. The models quickly reached their optimal configuration, dropping in loss in the first 10 epochs and then leveling out. The model that was optimized on the T2L-C variant was the most accurate in terms of the RMSE metric, while the model for the T2L-CO variant was most accurate in terms of the MAE metric. However, the differences are close to negligible.

Average Minimum Distances

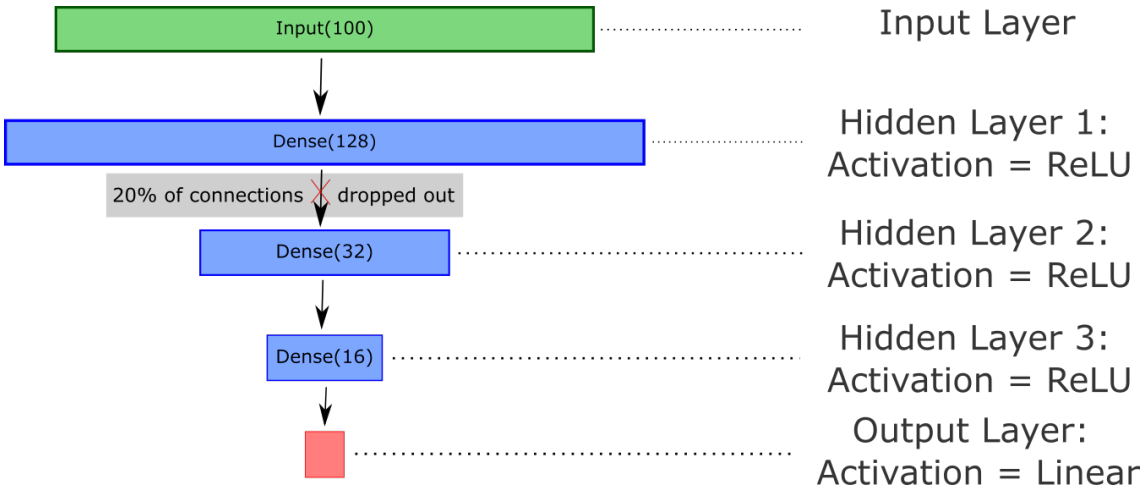


Figure 17: Optimal Dense Neural Network Architecture for AMDs, both the T2L and the T2L-CON variant. [Visualized with Inkscape]

As shown in Table 4, the first 100 AMD values of the T2L variant of the crystal structures were scaled with MinMax scaling (Equation 9), but not L2-normalized, in order to achieve optimal results. The labels underwent no preprocessing steps. The architecture of the dense neural network is shown in Figure 17. It includes the

[34]github.com/JRopes/CrystalEnergyPrediction/blob/main/DF_T2L_CO_DNN_Predictor.ipynb

input layer for the 100 features, followed by the first densely connected layer with 128 neurons, which is activated by a ReLU function. The first hidden layer is then followed by another densely connected layer, consisting of 32 neurons, which is also activated by a ReLU function. 20% of the connections to this layer are dropped out in order to avoid overfitting. The third and final hidden layer is densely connected without any dropout. It consists of 16 neurons and is activated by a ReLU function. Finally, a dense connection to single neuron is activated by a linear function to work as the output layer. This provides the final output as a scalar value. As the labels were not scaled in the preprocessing step, this value represents the predicted crystal energy and does not require an inverse scaling transformation. The implementation of this network can be found on GitHub[35].
The model, optimized on the T2L-CON variant of the AMDs, performed slightly worse but made use of the same preprocessing steps. As the first 100 AMDs were used, much like the network for the T2L variant, the same network architecture deemed optimal results. The implementation of this model can be found on GitHub[36]. Overall, both architectures have a total of 17,601 trainable parameters, significantly less than for the density functions. A high number of epochs were set and an early stopping callback function with a sensitivity of 20 epochs was applied in order to stop the training as soon as the loss no longer decreases, adjusting the weights to those of the epoch with the lowest loss. Finally, the training data was fed forward through the network in batches of 128 instances in order to make optimal weight-updates. With a learning rate of 0.01, the models' losses quickly drop within the first 10 epochs and then flatten. The learning curve on the training and validation data showed no signs of over- or underfitting and seemed to be an optimal final model for the energy prediction with AMD values.

[35] github.com/JRopes/CrystalEnergyPrediction/blob/main/AMD_T2L_DNN_Predictor.ipynb
[36] github.com/JRopes/CrystalEnergyPrediction/blob/main/AMD_T2L_CON_DNN_Predictor.ipynb

6 Final Results

6.1 Quantitative Results

Table 5: Final Results of the optimized models for each variant of AMDs and DFs, averaged over 10 experimental runs with 90% training data and 10% test data.

Input (Invariant)	Algorithm	Result Metrics (over 10 Runs)		
		RMSE$\pm std$	MAE$\pm std$	MAPE$\pm std$
AMDs (T2L)	Random Forest Regression	7.036 $\pm$ 0.230	5.476 $\pm$ 0.184	3.905 $\pm$ 0.117
	Gaussian Process Regression	6.309 $\pm$ 0.310	4.733 $\pm$ 0.223	3.390 $\pm$ 0.153
	Dense Neural Network	7.310 $\pm$ 0.176	5.587 $\pm$ 0.115	3.964 $\pm$ 0.088
AMDs (T2L-CON)	Random Forest Regression	7.813 $\pm$ 0.250	6.078 $\pm$ 0.240	4.376 $\pm$ 0.129
	Gaussian Process Regression	7.226 $\pm$ 0.197	5.513 $\pm$ 0.160	3.947 $\pm$ 0.111
	Dense Neural Network	8.233 $\pm$ 0.572	6.349 $\pm$ 0.414	4.511 $\pm$ 0.268
DFs (T2L-C)	Random Forest Regression	9.293 $\pm$ 0.265	7.224 $\pm$ 0.173	5.161 $\pm$ 0.118
	Gaussian Process Regression	9.467 $\pm$ 0.474	7.263 $\pm$ 0.394	5.183 $\pm$0.266
	Dense Neural Network	12.550 $\pm$ 0.488	9.430 $\pm$ 0.280	6.513 $\pm$ 0.158
DFs (T2L-CO)	Random Forest Regression	9.914 $\pm$ 0.530	7.564 $\pm$ 0.277	5.378 $\pm$ 0.181
	Gaussian Process Regression	9.586 $\pm$ 0.579	7.209 $\pm$ 0.277	5.135 $\pm$ 0.170
	Dense Neural Network	12.586 $\pm$ 0.620	9.317 $\pm$ 0.405	6.441 $\pm$ 0.244

The final results of this project are presented in Table 5. Each invariant, in combination with its optimized algorithm, was evaluated over 10 runs on 90% training data and 10% test data. The RMSE (Equation 6), MAE (Equation 4), and the MAPE (Equation 5) were used to provide an indicator of the performance of each model. A low RMSE value was prioritized, as a consistent low-error prediction is preferred over a model that has a very wide distribution of absolute errors on its predictions. All uncertainties are given as the standard deviation.

Both in terms of RMSE and MAE, the Gaussian Process Regression delivered the best results for the T2L variant of the AMDs, making predictions with a MAE of 4.733 $kJmol^{-1}$, on average. This means that the predictions with the Gaussian process were incorrect by only 3.390%, relative to the true crystal energy. The random forest takes the second place in terms of RMSE and MAE for the T2L AMDs. The optimized model's MAE is around 0.743 $kJmol^{-1}$ higher than for the Gaussian Process. Finally, the worst performing machine learning method for the T2L AMDs was the dense neural network, evaluating as the largest RMSE and MAE out of the three algorithms.

Regarding the T2L-CON variant of the AMDs, the ranking of the algorithms is identical. The best performance was achieved by the Gaussian process regression, evaluating with a RMSE of around 7.226 and a MAE of 5.513 $kJmol^{-1}$. This is followed by the random forest, having achieved a MAE of 0.565 $kJmol^{-1}$ higher than the Gaussian process. Finally, the worst performance on the T2L-CON variant came from the dense neural network, evaluating at a RMSE of 8.233 and a MAE of 6.349 $kJmol^{-1}$ with extremely high standard deviations, indicating its inconsistency.

Despite this variant ignoring the oxygen atom in the T2 molecule, the T2L-C variant of the DFs surprisingly showed better results than the T2L-CO variant in terms of the RMSE metric. The random forest performed best on this dataset, achieving

a RMSE of 9.293 and MAE of 7.224 $kJmol^{-1}$. This is followed by the Gaussian Process , scoring 9.467 on the RMSE and 7.263 $kJmol^{-1}$ on the MAE, which is only slightly worse than the random forest. The dense neural network, optimized on the T2L-C variant, performed the worst for this variant by a significant margin. It achieved a RMSE of 12.550 and a MAE of 9.430 $kJmol^{-1}$, a difference of 3.257 and 2.206 $kJmol^{-1}$ on the RMSE and MAE, respectively, in comparison to the Gaussian process.
Finally, for the T2L-CO variation of the DFs, the Gaussian process performed best out of the three algorithms by only a small margin. It averaged a RMSE of 9.586 and a MAE of 7.209 $kJmol^{-1}$. This means that the best performing algorithm for the T2L-CO variant scores a worse RMSE than the best performing algorithm for the T2L-C variant, but a slightly better MAE by a margin of 0.015 $kJmol^{-1}$. However, the relevance of this is questionable due to the nearly negligible margin that makes up only a small fraction of a single standard deviation of the metric. The Random Forest takes the second place for the T2L-CO DFs, in terms of RMSE and MAE. The optimized model made predictions with a RMSE of 9.914 and a MAE of 7.564 $kJmol^{-1}$. Finally, the dense neural network performed worst for the T2L-CO DFs and the overall project, with a significantly higher RMSE and MAE metric. The trained model achieved an MAE of 9.317 $kJmol^{-1}$ and an RMSE of 12.586.

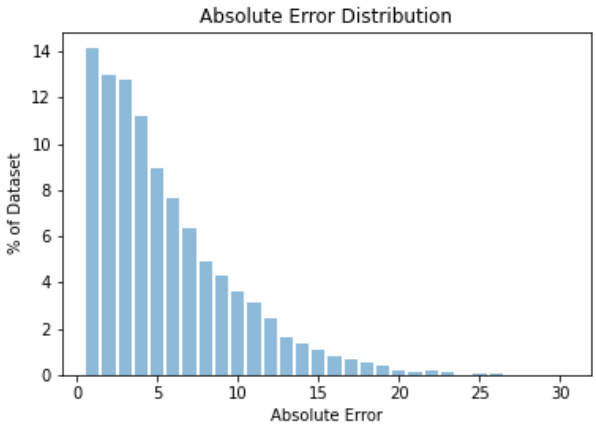


Figure 18: The distribution of error for up to 30 $kJmol^{-1}$, averaged over 10 runs, for the Gaussian Process with the T2L variant of the Average Minimum Distances. [Visualized with Matplotlib]

Shown in Figure 18 is the absolute error distribution of the Gaussian process, using the T2L variant of the AMDs; the model with the best overall performance in this project. The histogram is split up into bins of 1 $kJmol^{-1}$ and each bar represents the percentage of instances that were predicted with an absolute error of less than the corresponding threshold. This distribution is averaged over a total of 10 runs. According to these results, $\sim$ 51% of the instances were predicted with an absolute error of less than 4 $kJmol^{-1}$. The remaining instances were predicted with absolute errors of up to 26 $kJmol^{-1}$, although the majority of them with an absolute error of less than 8 $kJmol^{-1}$. The industry target of sub-$kJmol^{-1}$ predictions was met with an average of 14% of the predictions.

6.2 Computational Speeds

An important factor in CSP, specifically energy ranking, is the efficiency of the algorithms. The computation times of the various algorithms and invariants are shown below in Table 6. These benchmarks come from an Intel Xeon CPU, clocked at 2.3 GHz. The algorithms were trained on 90% of the data, 5111 instances, and evaluated on a total of 10% of the data, 568 instances. The uncertainty of the times is not given by the standard deviation, but rather the observed range in the experiments.

Table 6: Computation Times of the various optimized algorithms for each variant of the AMDs and DFs. This timedata is recorded on an Intel Xeon CPU, clocked at 2.3 GHz.

Input (Invariant)	Algorithm	Training Time (s)	Prediction Time (ms)	Prediction Time per Instance (ms)
AMDs (T2L)	Random Forest Regression	62 $\pm$ 1	108 $\pm$ 3	~0.19
	Gaussian Process Regression	451 $\pm$ 59	9 984 $\pm$ 85	~17.58
	Dense Neural Network	17 $\pm$ 2	125 $\pm$ 5	~0.22
AMDs (T2L-CON)	Random Forest Regression	65 $\pm$ 1	130 $\pm$ 54	~0.23
	Gaussian Process Regression	517 $\pm$ 28	9 068 $\pm$ 1 440	~15.96
	Dense Neural Network	11 $\pm$ 5	107 $\pm$ 5	~0.19
DFs (T2L-C)	Random Forest Regression	2 969 $\pm$ 477	211 $\pm$ 5	~0.37
	Gaussian Process Regression	1 675 $\pm$ 620	10 240 $\pm$ 1 755	~18.03
	Dense Neural Network	154 $\pm$ 30	336 $\pm$ 22	~0.59
DFs (T2L-CO)	Random Forest Regression	1 223 $\pm$ 10	211 $\pm$ 5	~0.37
	Gaussian Process Regression	1 456 $\pm$ 195	6 454 $\pm$ 390	~11.36
	Dense Neural Network	38 $\pm$ 14	226 $\pm$ 46	~0.40

Generally, the Gaussian process was the slowest algorithm by significant margins, both in terms of training and predicting. However, while this algorithm may be the slowest, it is still relatively fast in comparison to other prediction methods, such as DFT+MBD, taking only around 451 seconds to train on the T2L AMD variant with the first 100 AMDs and making predictions in around 17.58 milliseconds per instance [8]. The random forest and the dense neural network tend to both be much faster than the Gaussian process in terms of computation times, for both training and prediction. While they made predictions at similar speeds, the dense neural network was the fastest algorithm in this project in terms of training time. When observing the training times on the first 100 values of the T2L AMD variant, the random forest finished training after around 62 seconds while the dense neural network took only 17 seconds, a difference by a factor of around three. In terms of prediction, the two algorithms reached extremely fast speeds, both in the sub-ms space for a single instance.

When comparing the use of DFs and AMDs, the differences in the computational efficiency were immense. On average, the training of the T2L AMD on 5111 instances was 8 times faster than the use of the T2L-CO DF, which is the fastest variant of the DFs. Although the prediction speeds were all within a few milliseconds per instance,

the DFs were significantly slower in terms of training, especially when considering that the computation of the DFs is extremely computationally intensive in comparison to that of the AMD values. Therefore, the use of the Gaussian process with the T2L AMD dataset is optimal, not only in terms of its prediction accuracy but also its speed. The computation of the first 100 AMD values took only a few ms per instance, the Gaussian process fit this data within 7.5 $minutes$ for 5111 instances, and it achieved a MAE of 4.733 $kJmol^{-1}$ with a prediction speed of around 17.58 ms per instance [18, 32].

6.3 Evaluation

Overall, the primary goal of making predictions of the crystal energy with a mean absolute error of less than 4 $kJmol^{-1}$ was not achieved. However, the use of AMDs, specifically the T2L variant, with the Gaussian process regression came close to achieving this goal by around 0.733 $kJmol^{-1}$. The experiments did not only provide a proof of concept for fast crystal energy prediction with isometry invariants, but also provided a comprehensive comparison between the effectiveness of AMDs and DFs of crystal structures, as well as the use of various machine learning algorithms, including random forests, Gaussian processes, and dense neural networks.
In regards to the invariants, the difference between the prediction results, as shown in Table 5, is clear. The use of AMDs as input resulted in significantly more accurate predictions, with a difference in MAE of around 2.476 $kJmol^{-1}$ between the best performances on AMDs and DFs. All models using any of the variants of the AMDs outperformed any of the models using the two variants of the DFs. In addition, as only the first 100 AMDs were used compared to at least 936 features of the T2L-CO dataset, the algorithms were able to achieve the results with significantly less training time. The difference of 1005 s (just under 17 minutes) on the training times of the Gaussian process using the T2L variant of the AMDs and the T2L-CO variant of the DFs (the variant with the least features of the two DF variants) makes this very apparent. In addition, the advantages of using AMDs is especially noticeable when comparing the optimized dense neural network architectures. The network for the AMDs has a total of 17,601 trainable parameters while the network for the T2L-CO variant of the DFs has a massive 1,028,353 trainable parameters, a difference by a factor of 58. In essence, the use of DFs is more computationally expensive, both in computing the invariant and training the models, and delivers significantly worse results. One explanation for this could be the large percentage of zeros in the feature vector of a DF. As the DFs of two different structures have different domains, the largest domain in the T2 dataset must be used in order to consistently align all k DFs when serializing them into a single feature vector. This results in a feature vector of largely zeros. Better preprocessing methods for DFs could possibly improve the prediction performance. However, such methods were not discovered in the scope of this project.
Due to the use of two variants of DFs, an interesting comparison can be made between the two. The T2L-C variant used a sample rate of 10 but did not consider the three oxygen atoms. The T2L-CO variant had a lower sample rate of 5 but considered the entire crystal structure. Primarily, the difference in computation times between the two variants is significant, both in terms of training and prediction

time, and is especially noticeable for the random forest. The models for the T2L-CO variant were able to train faster, on average, and make predictions with at least the same speed. In addition, it was hypothesized that the T2L-CO variant would hold more information on the crystal structure and, therefore, have a better performance. However, the T2L-CO variant performed slightly worse than the T2L-C variant in terms of the prioritized RMSE metric, but slightly better in terms of the MAE, although by only a small margin. While the T2L-CO variant provided more detail on the crystal structure in theory, its lower sample rate could have potentially caused worse results in terms of consistently accurate predictions. This means that by increasing the sample rate of the DFs, a better peformance might be achieved. However, the computation of the invariant would then become even more expensive than it already is, ultimately speaking for AMDs as the preferred isometry invariant for crystal energy prediction. It required smaller models with shorter training times, is less expensive to compute, and delivers superior performance.

Similarly, the use of two variants of AMDs can provide results on whether the lack of using the small hydrogen atoms in the computation of the T2L-CON dataset actually provided better results on crystal energy prediction, as hypothesized. While the general hypothesis of the small hydrogen atoms having little impact on the crystal energy seems reasonable, the results show that the use of the T2L dataset variant, including all atoms, showed significantly better results by a margin of 0.780 $kJmol^{-1}$ on the MAE. This can be explained by the fact that even small atoms still have an impact on the crystal structure and energy, despite it potentially being minimal. In addition, and unlike DFs, the differences in training and prediction speeds were much less severe. However, this was expected as similar feature sizes were used for both variants, unlike for DFs.

The performances of the various models show some interesting insights. For both variants of the AMDs, the Gaussian process regression performed best, followed by the dense neural network and the random forest. For the T2L-C DFs, the random forest performed best, followed by the Gaussian process and the dense neural network. Finally, for the T2L-CO DFs, the Gaussian process performed best again, followed by the random forest and the dense neural network. This shows that there is no consistently superior model for the general crystal energy prediction on isometry invariants, and depending on the invariant, different models prove to perform better than others. However, the Gaussian process showed the best performance on three of the four variants and achieved the best results for the entire project. This shows that the Gaussian process is an effective approach to crystal energy prediction with isometry invariants, both in terms of speed and accuracy.

Overall, the results show that the use of the first 100 AMDs of the T2L variant with the Gaussian process regression provided the best performance in crystal energy prediction. The AMDs were the least expensive to compute, especially since only the first 100 values were used, and require much smaller models, resulting in shorter training and prediction times. Although the Gaussian process is still the slowest algorithm out of the three, its training and prediction times were very fast in comparison to previous state-of-theart methods. The Gaussian process is simple in training and optimization and is unable to overfit on the training data. It does not only output a predicted scalar value, but also the associated standard deviation, which can be used to calculate the confidence interval for the prediction. In comparison to the current state-of-the-art DFT+MBD approach, taking multiple hours

for a prediction, the Gaussian process is significantly faster,only taking a few milliseconds per instance, while coming close to the same accuracy by a factor of 2.5 [8].

7 Discussion

7.1 Conclusion

Within the scope of this project, an exploration of the feasibility of using isometry invariants as an input to machine learning methods to predict the crystal energy of crystal structures was presented. Overall, it was shown that the use of AMDs and DFs of crystal structures, modelled as point sets, is a viable and efficient input to the regression problem, introducing great computational efficiency while still reaching an adequate prediction accuracy. In addition, various machine learning methods were compared and evaluated, ultimately leading to the application of the Gaussian process with a rational quadratic kernel on the T2L variant of the AMDs.
As the original use for geometric invariants in crystal energy prediction was unknown, the results of this research project have proven their potential, achieving results that are significantly better than random predictions and, therefore, indicating a clear correlation between the two isometry invariants and the crystal energy. As presented in Section 6.1, the use of AMDs and DFs achieved MAEs of 4.733 ± 0.223 $kJmol^{-1}$ and 7.209 ± 0.533 $kJmol^{-1}$, respectively. With the crystal energy range of the entire T2 dataset being around 100 $kJmol^{-1}$, predictions within 10% of that are significant and further work on this could improve the overall prediction approach. Although the industry target of a sub-$kJmol^{-1}$ accuracy was not reached, this was not expected initially but rather hoped for.
In addition, various machine learning methods were optimized, experimented with, and compared on the invariants as input. Random forest regression, Gaussian process regression, and dense neural networks showed promising, yet unexpected results. Although it was hypothesized that dense neural networks would show the best performance, Gaussian processes proved to be superior by significant margins. The use of the first 100 T2L AMDs as input to the Gaussian process regression with a rational quadratic kernel did not only provide the best result of the entire project, but introduced many other benefits, such as its inability to overfit and its prediction with confidence intervals. The computational speed of the presented algorithms is apparent, especially when compared to the state-of-the-art DFT+MBD method, which takes hours for a single prediction.
Overall, this research project proposed the use of probabilistic machine learning methods and isometry invariants as a computationally efficient method for the energy ranking in CSP. The Gaussian process with the rational quadratic kernel did not only demonstrate relatively accurate prediction capabilities, but did so with simplicity and a high computational efficiency. Rather than making use of various descriptors and chemical properties of a crystal structure, the use and computation of AMDs is simple, solely focusing on the geometry of the structure. With a computation time of a few ms per crystal instance to accumulate the input data, a training time of a few minutes, and a prediction time of around 17.58 ms per instance, this method is significantly faster than previously introduced techniques of predicting the absolute crystal energy. Considering this project as a proof of concept, the con-

tinuation of research on the use of isometry invariants in crystal energy prediction can greatly contribute to CSP as a whole, ultimately solving the great efficiency problem.
Finally, referring back to Section 1.7, which introduced the aims and objectives of this project, not all of the initial targets were met as the completion of one objective was unsuccessful. All aims were accomplished, although this statement can only be made subjectively. A proof of concept for the crystal energy prediction with geometric invariants (A1) was arguably delivered in this project and various machine learning algorithms were explored (A2). The effectiveness of DFs and AMDs were compared (A3) and the completed research was presented in an appropriate format, comprehenisble to professionals with the relevant scientific background (A5). The report shows an understanding of the topic of crystallography (A5) and concluded by formalizing a procedure for predicting crystal energy based on geometric invariants (A6). In regards to the objectives, characterized by their objectivity and, therefore, measurability, all but one objective was met. The optimized settings for each invariant and algorithm were presented (O1), however, the implementation of an algorithm that predicts the crystal energy with an error of less than 4 $kJmol^{-1}$ was not successful (O2). Finally, the optimized Gaussian process was able to train on the T2 dataset in significantly less time than an hour (O3) and successfully predicted the crystal energy in a few seconds (O4). Overall, the majority of aims and objectives were met. The code for this project is available and openly accessible on GiHub[37].

7.2 Further Work

The presented findings enable the formulation of new questions and ideas for the continuation of this research. Isometry invariants as input to crystal energy prediction have been proven as a viable concept, especially in terms of the computational efficiency. However, further research on this is highly recommended. A limitation on this project is the dataset, consisting of a single compound and a limited number of instances. The T2 dataset consists of 5679 predicted arrangements for Benzimidazolone Triptycene. While this dataset allows the development of prediction algorithms for crystal energy, it does not allow the evaluation of how generally applicable these methods are. As the goal of crystal structure prediction is the ability to function on any compound, the T2 dataset, consisting of a single compound, limits the development and evaluation of this. Therefore, further work on the prediction with isometry invariants, using datasets of other compounds, is highly recommended in order to develop and evaluate the proposed methods for general use.
In addition, while 5679 data instances are a sufficient number for most machine learning methods, some state-of-the-art neural network architectures require much larger datasets. As introduced in Section 1.6, the deep residual neural network is an extremely promising approach for nonlinear regression problems, such as the prediction of crystal energy. The application of this architecture within the scope of this project was considered, even making a few prototyping attempts, but was significantly limited by the size of the dataset. The residual network, introduced by Dongwei Chen et al., is trained on a dataset, consisting of 6.75 million instances, to

[37] github.com/JRopes/CrystalEnergyPrediction

reach the high level of accuracy for nonlinear regression [29]. Therefore, while a deep residual neural network may be a powerful technique for crystal energy prediction using isometry invariants, the size of the T2 dataset limited the exploration of this method. Further work with larger datasets is recommended, considering the deep residual neural network as a method with great potential. Combining the two previously mentioned points, the accumulation of data to generate one large dataset, consisting of multiple compounds, would provide opportunities for the generalization of algorithms and the exploration of more complex architectures that require more training data.
In addition, this project, along with previous work introduced in Section 1.6, showed the great potential of Gaussian processes in crystal energy ranking. This probabilistic machine learning method should be pursued and further explored in future work. This should take place in the form of further experiments on various kernels, potentially even developing a customized kernel for the AMDs of crystal structures. As this project proposed a fast crystal energy prediction method with an adequate accuracy, future goals include the improvement of this accuracy. A primary factor in the accuracy is the kernel and there could be better options than the rational quadratic kernel, deeming optimal results within the scope of this project. An improvement in the kernel of this method could make the use of the Gaussian process a fast, simple, but also accurate method of prediction, potentially making it deployable in industry.
Finally, more complex and customized implementations of the Gaussian process should be experimented with in order to generally improve this method. While this project made use of the SciKit-Learn implementation, libraries such as GPyTorch[38] allow the increase in efficiency through GPU acceleration, the application of Gaussian processes to larger datasets through a scalable approach, and the implementation of highly customized Gaussian processes with a modular design [59]. This could further improveme the already established computational speed with AMDs and the scaling of this method, enabling its application on large datasets with millions of data points [59]. In addition, using the modular design, new architectures can be experimented with. This includes more complex models such as a Deep Gaussian process, a "deep belief network based on Gaussian process mappings", which could potentially further improve the accuracy of this probabilistic machine learning method [60].

8 BCS Criteria & Self-Reflection

As expected by the Chartered Institute of IT[39], the six outcomes for the final year project are met. These outcomes include:

1. An ability to apply practical and analytical skills gained during the degree programme.

2. Innovation and/or creativity.

[38] gpytorch.ai/
[39] See here: www.bcs.org/

3. Synthesis of information, ideas and practices to provide a quality solution together with an evaluation of that solution.

4. That the project meets a real need in a wider context.

5. An ability to self-manage a significant piece of work.

6. Critical self-evaluation of the process.

Outcome 1 is shown throughout the entire project, applying data preprocessing techniques, machine learning methods, and evaluation skills to work on the problem of crystal energy prediction. I analysed the original problem, classifying it as a regression problem, and familiarized myself with the input and dataset that I would be using. I decided on appropriate preprocessing methods for the data and selected machine learning methods that have potential for this problem. Finally, I experimented with various preprocessing methods, hyperparameters, and architectures in experiments and evaluated their final results in order to draw conclusions on the optimal approach. Outcome 2 is mentioned in Section 1.1, outlining the significance of this research topic. I explored a new approach of crystal energy prediction, using isometry invariants, rather than other properties of crystal structures. I came up with viable solutions with adequate accuracy and exceptional speeds, providing a proof of concept for using Gaussian processes on isometry invariants to predict crystal energy. The report of this research project meets Outcome 3, having closely considered previous work on the topic, modern machine learning methods, and scientific practices to come up with a solution and synthesize a scientific report on the topic. This required the in-depth understanding of the topic of crystal energy prediction, developed by reading a significant amount of previous research. As mentioned in Section 1.1, Outcome 4 is met as the topic of this project focuses on currently relevant research that is slowly moving into industrial practice. The problem of predicting crystal structures on the atomic level is considered a fundamental challenge in material sciences and crystal energy ranking is one aspect of that [11]. Therefore, exploring the use of new methods for this process, such as using isometry invariants, is an extremely relevant topic and meets a real need in the field of material sciences, potentially enabling the discovery of new pharmaceutical compounds and various other materials.
Outcome 5 proved to be one of the most difficult aspects of this final year project. Throughout the year we had only been given three deadlines and had to manage the time for these, which was quite difficult, especially with the general lack of a routine due to the pandemic. Compared to the original project plan that I introduced in the proposal, I believe that I managed to follow the schedule quite well, never entering a situation in which I was under a significant amount of stress. In terms of general management of the project, I believe that I had a clear idea of what was expected of me and what I expected of myself, and therefore managed to produce a piece of work that I am proud of. The CRISP-DM development process, as introduced in Section 4.2, helped me with managing the research, development, and assessment for this project, iteratively experimenting with prototypes and evaluating them to decide on further steps. Regarding the actual report, I started writing this quite early and adding sections throughout the entire duration of the project. I managed this writing process efficiently by writing a very specific outline at the beginning of the project. This helped me envision and set the goals for the final report and

allowed me to easily identify sections that I could already work on and complete. Finally, I started using a planner in which I entered all my submission deadlines, further splitting these into personal deadlines for myself, allowing me to stay on track. This enabled me to plan out my work for every single day in order to succeed in my modules while also putting a constant effort into the synthesis of my final year project. The general approach of investing a lot of time into research on previous related work early on in the project helped me significantly in being able to plan and mange my work, as it gave me a realistic idea of the tasks that require completion in order to meet all the objectives of the project.

In order to meet Outcome 6, and for a general interest in improving myself, it is important to be critical of your own work. There are several aspects of this project that I believe I could improve in. First, while I did start early with writing the project report, I should have also started earlier with my final version. I only completed a full draft less than 2 weeks before the final deadline and believe that if I had this full draft even earlier, more feedback could have been obtained to perfect the format and formulation of this report. In addition, I also believe that I should have computed more of my data right at the beginning of this dissertation. Especially the density functions were computed quite late and required two variants made up of 5679 instances each. As this process is computationally intensive (multiple days of CPU time for the T2 dataset), it always put a stop into my work. For example, while I was doing experiments using the T2L-C variant of the density functions, I realized that I was not getting satisfactory results. I only then realized that I did not include the 3 oxygen atoms and then again spent multiple days computing the new T2L-CO variant. If I had just spent a few weeks right at the beginning of the project on computing these datasets, I could have had a more continuous development process. Within the topic of data, I could have applied better data collection. Instead of saving all the predictions for each experiment, I only calculated the average of the accuracy metrics. Later in the project, I realized that other metrics such as the R^2 metric could have been very useful, but would have meant the repetition of all experiments. Had I saved all the data in a CSV file, I could have quickly written a program that calculates any necessary metrics on this data without redoing the experiments. Regarding the dense neural network, I believe that more research on more complex network architectures could have been completed, allowing a better performance of this method. Deep learning is currently a very popular method and my implementation of the dense neural network is somewhat underwhelming. More research on this and additional experimentation with unique architectures could have potentially improved the results. However, the purpose of this project was not to obtain a single complex implementation, but rather the comparison of multiple methods in order to find the optimal algorithm for isometry invariants. Finally, more frequent meetings could have greatly contributed to the direction of my project. I met with Dr. Kurlin (supervisor) for bigger milestones, while meeting with Philip Smith (PhD student supervisor) more frequently to discuss my process and plans. However, this quickly evolved into pure email communication and while this still allowed frequent interaction about the process, I believe that more physical meetings could have pushed me to engage in discussions and pitch more ideas to develop the findings into a more mature method. Overall, I believe my approach for this final year project worked very well for me and the resulting implementation and report does not only meet the six outcomes of the Chartered Institute of IT, but also my

own expectations.

9 References

[1] Sharmistha datta and David J. W. Grant. "Crystal Structures of Drugs: Advances in Determination, Prediction and Engineering". In: *Nature Reviews Drug Discovery* 3 (2004), pp. 42–57. DOI: 10.1038/nrd1280.

[2] Sarah L. Price, Doris E. Braun, and Susan M. Reutzel-Edens. "Can computed crystal energy landscapes help understand pharmaceutical solids?" In: *Chemical Communications* 52.44 (2016), pp. 7065–7077. ISSN: 1364548X. DOI: 10.1039/c6cc00721j.

[3] Sarah L. Price. "From crystal structure prediction to polymorph prediction: interpreting the crystal energy landscape". In: *Physical Chemistry Chemical Physics* 10.15 (2008), pp. 1996–2009. ISSN: 14639076. DOI: 10.1039/b719351c.

[4] "Guideline on the chemistry of active substances". In: *Chemistry of active substances*. European Medicines Agency, 2017. URL: https://www.ema.europa.eu/en/chemistry-active-substances-chemistry-new-active-substances.

[5] Pawanpreet Sing and Renu Chadha. "Crystal Structure Prediction in the Context of Pharmaceutical Polymorph Screening and Putative Polymorphs of Ciprofloxacin". In: *International Journal of Pharmacy and Pharmaceutical Sciences* 9.4 (2017). ISSN: 09751491. DOI: 10.22159/ijpps.2017v9i4.14332.

[6] Jonas Nyman and Susan M. Reutzel-Edens. "Crystal structure prediction is changing from basic science to applied technology". In: *Faraday Discussions* 211 (2018), pp. 459–476. DOI: 10.1039/C8FD00033F.

[7] David Taylor. "The Pharmaceutical Industry and the Future of Drug Development". In: *Pharmaceuticals in the Environment*. The Royal Society of Chemistry, 2016, pp. 1–33. ISBN: 978-1-78262-189-8. DOI: 10.1039/9781782622345-00001. URL: http://dx.doi.org/10.1039/9781782622345-00001.

[8] Johannes Hoja and Alexandre Tkatchenko. "First principle stability ranking of molecular crystal polymorphs with the DFT+MBD approach". In: *Faraday Discussions* 211 (2018), pp. 253–274. ISSN: 13645498. DOI: 10.1039/c8fd00066b.

[9] Felix Musil et al. "Machine learning for the structure-energy-property landscape of molecular crystals". In: *Journal of Chemical Science* 9.5 (2018), pp. 1289–1300. ISSN: 20416539. DOI: 10.1039/c7sc04665k.

[10] Anthony M. Reilly et al. "Report on the sixth blind test of organic crystal structure prediction methods". In: *Acta Crystallographica Section B* 72.4 (2016), pp. 439–459. ISSN: 13645498. DOI: 10.1107/S2052520616007447.

[11] Scott M. Woodley and Richard Catlow. "Crystal structure prediction from first principles". In: *Nature Mater* 7.12 (2008), pp. 937–946. DOI: 10.1038/nmat2321.

[12] Gautam Desiraju. "Cryptic crystallography". In: *Nature Materials* 1.2 (2002), pp. 77–79. DOI: 10.1038/nmat726.

[13] Johannes Hoja et al. "Reliable and Practical Computational Prediction of Molecular Crystal Polymorphs". In: *Science Advances* 5.1 (2019). DOI: 10.1126/sciadv.aau3338.

[14] Anthony R. West. *Basic Solid State Chemistry*. 2nd ed. Wiley, 1999. ISBN: 0471987557.

[15] Massimo Nespolo. "Lattice versus structure, dimensionality versus periodicity: a crystallographic Babel?" In: *Journal of Applied crystallography* 52.2 (2019), pp. 451–456. DOI: 10.1107/S1600576719000463.

[16] Olga Anosova and Vitaliy Kurlin. *Introduction to Periodic Geometry and Topology*. 2021. arXiv: 2103.02749 [cs.CG].

[17] Howard hiller. "Crystallography and Cohomology of Groups". In: *The American Mathematical Monthly* 93.10 (1986), pp. 765–779. DOI: 10.2307/2322930.

[18] Daniel Widdowson et al. "Average Minimum Distances of periodic point sets". In: (2020). arXiv: 2009.02488 [cond-mat.mtrl-sci].

[19] Herbert Edelsbrunner et al. "The Density Fingerprint of a Periodic Point Set". In: *Proceedings of SoCG (2021)* (2021). arXiv: 2009.02488 [cs.CG].

[20] Paul J. Olver. *Equivalence, Invariants, and Symmetry*. 1st ed. Cambridge University Press, 1995. ISBN: 1316583732.

[21] A. F. Kapustinskii. "Lattice Energy of Ionic Crystals". In: *Quarterly Reviews, Chemical Society* 10.3 (1956). DOI: 10.1039/QR9561000283.

[22] Angeles Pulido et al. "Functional materials discovery using energy - structure - function maps". In: *Nature* 543 (2017), pp. 657–664. ISSN: 14764687. DOI: 10.1038/nature21419.

[23] Artem Oganov et al. "Structure prediction drives materials discovery". In: *Nature Reviews Materials* 4.5 (2019), pp. 331–348. DOI: 10.1038/s41578-019-0101-8.

[24] Sarah L. Price. "Is zeroth order crystal strucute prediction (CSP_0) coming to maturity? What should we aim for in an ideal crystal structure prediction code?" In: *Faraday Discussions* 211 (2018), pp. 9–30. DOI: 10.1039/C8FD00121A.

[25] Andrei V. Kazantsev et al. "Successful prediction of a model pharmaceutical in the fifth blind test of crystal structure prediction". In: *International Journal of Pharmaceutics* 418.2 (2011). A priori Performance Predictions, pp. 168–178. ISSN: 0378-5173. DOI: 10.1016/j.ijpharm.2011.03.058. URL: https://www.sciencedirect.com/science/article/pii/S0378517311002948.

[26] Denis Quane. "Crystal Lattice Energy and the Madelung Constant". In: *Journal of Chemical Education* 47.5 (1970), pp. 396–398. DOI: 10.1021/ed047p396.

[27] Sandip De et al. "Comparing molecules and solids across structural and alchemical space". In: *Phys. Chem. Chem. Phys.* 18 (20 2016), pp. 13754–13769. DOI: 10.1039/C6CP00415F.

[28] Olga Egorova et al. "Multi-fidelity Statistical Machine Learning for Molecular Crystal Structure Prediction". In: (2020). DOI: 10.26434/chemrxiv.12407831.v1. URL: https://chemrxiv.org/articles/preprint/Multi-fidelity_Statistical_Machine_Learning_for_Molecular_Crystal_Structure_Prediction/12407831.

[29] Dongwei Chen et al. "Deep Residual Learning for Nonlinear Regression". In: *Entropy* 22.2 (2020). DOI: 10.3390/e22020193.

[30] Sydney R. Hall, Frank H. Allen, and I. David Brown. "The crystallographic information file (CIF): a new standard archive file for crystallography". In: *Acta Crystallographica Section A* 47.6 (1991). DOI: 10.1107/S010876739101067X.

[31] Philip Smith. *Density Functions in C++*. https://github.com/Phil-Smith1/Density_Functions. 2021.

[32] Daniel Widdowson et al. *The asymptotic behaviour and a near linear time algorithm for isometry invariants of periodic sets*. 2021. arXiv: 2009.02488 [cond-mat.mtrl-sci].

[33] Marco Mosca. *Average Minimum Distances in C++*. https://github.com/mmmosca/AMD. 2021.

[34] Daniel Widdowson. *Average Minimum Distances in Python*. https://github.com/dwiddo/AMD. 2021.

[35] Clare F. Macrae et al. "*Mercury 4.0*: from visualization to analysis, design and prediction". In: *Journal of Applied Crystallography* 53.1 (Feb. 2020), pp. 226–235. DOI: 10.1107/S1600576719014092. URL: https://doi.org/10.1107/S1600576719014092.

[36] *Inkscape*. https://inkscape.org/.

[37] François Chollet et al. *Keras*. https://keras.io. 2015.

[38] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[39] J. Bergstra, D. Yamins, and D. D. Cox. "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures". In: *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)* (2013).

[40] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[41] The pandas development team. *pandas-dev/pandas: Pandas*. Version 1.1.5. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[42] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[43] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[44] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[45] Jui-Chan huang et al. "Application and comparison of several machine learning algorithms and their integration models in regression problems". In: *Neural Computing and Applications* 32.10 (2020), pp. 5461–5469. ISSN: 14333058. DOI: 10.1007/s00521-019-04644-5.

[46] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033295X. DOI: 10.1037/h0042519.

[47] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 1st ed. MIT Press, 2016. ISBN: 9780262035613.

[48] H. Leung and S. Haykin. "The complex backpropagation algorithm". In: *IEEE Transactions on Signal Processing* 39.9 (1991), pp. 2101–2104. ISSN: 19410476. DOI: 10.1109/78.134446.

[49] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *The 3rd International Conference for Learning Representations 2015* (2017). arXiv: 1412.6980.

[50] Andrew Gordon Wilson, David A. Knowles, and Zoubin Ghahramani. "Gaussian Process Regression Networks". In: *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)* 1 (2012), pp. 599–606.

[51] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. 1st ed. MIT Press, 2006. ISBN: 9780262182539.

[52] Anthony J. Myles et al. "An introduction to decision tree modeling". In: *Journal of Chemometrics* 18.6 (2004), pp. 275–285. DOI: 10.1002/cem.873.

[53] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[54] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24 (1996), pp. 123–140. DOI: 10.1007/BF00058655.

[55] Khaled Fawagreh, Mohamed Medhat Gaber, and Eyad Elyan. "Random forests: from early developments to recent advancements". In: *Systems Science Control Engineering* 2.1 (2014), pp. 602–609. DOI: 10.1080/21642583.2014.956265.

[56] S. Gopal Krishna Patro and Kishore Kumar Sahu. *Normalization: A Preprocessing Stage*. 2015. arXiv: 1503.06462.

[57] Rüdiger Wirth and Jochen Hipp. "CRISP DM: Towards a Standard Process Model for Data Mining". In: *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining* (2000), pp. 29–39.

[58] Gregory P. Meyer. *An Alternative Probabilistic Interpretation of the Huber Loss*. 2020. arXiv: 1911.02088.

[59] Jacob R Gardner et al. "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration". In: *Advances in Neural Information Processing Systems*. 2018.

[60] Andreas C. Damianou and Neil D. Lawrence. *Deep Gaussian Processes*. 2013. arXiv: 1211.0358 [stat.ML].

10 Appendices

APPENDIX A: Kernel Experiments for Gaussian Process with Average Minimum Distances

Kernel	Dataset Variant	Feature Size	Result Metrics			Computation Time		
			RMSE±*std*	MAE±*std*	MAPE±*std*	Total Training±*std* (s)	Total Prediction±*std* (ms)	Prediction per Instance ±*std* (ms)
Rational Quadratic	*T2L-CON*	*50*	7.364 ± 0.201	5.637 ± 0.146	4.032 ± 0.095	603 ± 82	15568 ± 191	13.7 ± 0.2
		100	7.358 ± 0.153	5.637 ± 0.116	4.039 ± 0.080	341 ± 50	8379 ± 928	7.4 ± 0.8
		150	7.205 ± 0.133	5.502 ± 0.101	3.940 ± 0.067	592 ± 62	15940 ± 85	14.0 ± 0.1
		200	7.261 ± 0.102	5.546 ± 0.108	3.974 ± 0.075	516 ± 49	14191 ± 103	12.5 ± 0.1
		250	7.304 ± 0.127	5.564 ± 0.115	3.979 ± 0.082	565 ± 32	16294 ± 194	14.3 ± 0.2
		300	7.274 ± 0.209	5.605 ± 0.167	4.018 ± 0.123	512 ± 46	14441 ± 162	12.7 ± 0.1
		350	7.270 ± 0.163	5.575 ± 0.125	3.995 ± 0.101	524 ± 61	12926 ± 34	11.4 ± 0.1
		400	7.282 ± 0.156	5.572 ± 0.104	3.986 ± 0.079	643 ± 27	17644 ± 60	15.5 ± 0.1
		450	7.343 ± 0.143	5.640 ± 0.110	4.039 ± 0.075	593 ± 19	17759 ± 64	15.6 ± 0.1
		500	7.251 ± 0.147	5.584 ± 0.129	4.010 ± 0.099	563 ± 38	13826 ± 47	12.2 ± 0.1
	T2L	*50*	6.503 ± 0.123	4.900 ± 0.86	3.509 ± 0.059	627 ± 85	15961 ± 183	14.1 ± 0.2
		100	6.344 ± 0.152	4.801 ± 0.103	3.439 ± 0.070	349 ± 47	7979 ± 564	7.0 ± 0.5
		150	6.607 ± 0.119	4.977 ± 0.077	3.559 ± 0.053	400 ± 23	12789 ± 203	11.3 ± 0.2
		200	6.617 ± 0.147	4.966 ± 0.114	3.554 ± 0.079	506 ± 40	15943 ± 46	14.0 ± 0.1
		250	6.517 ± 0.109	4.914 ± 0.082	3.514 ± 0.055	574 ± 91	16464 ± 193	14.5 ± 0.2
		300	6.632 ± 0.139	5.003 ± 0.092	3.577 ± 0.062	545 ± 15	16431 ± 52	14.5 ± 0.1
		350	6.615 ± 0.077	4.990 ± 0.077	3.581 ± 0.053	500 ± 22	12395 ± 44	10.9 ± 0.1
		400	6.611 ± 0.149	4.984 ± 0.080	3.569 ± 0.053	585 ± 25	17906 ± 201	15.8 ± 0.2
		450	6.559 ± 0.179	4.954 ± 0.127	3.545 ± 0.085	512 ± 21	12927 ± 67	11.4 ± 0.1
		500	6.622 ± 0.116	5.004 ± 0.092	3.581 ± 0.068	598 ± 24	18429 ± 219	16.2 ± 0.2
Squared Exponential	*T2L-CON*	*50*	85.895 ± 0.181	85.036 ± 0.198	62.464 ± 0.214	355 ± 8	15595 ± 144	13.7 ± 0.1
		100	85.768 ± 0.268	84.897 ± 0.305	62.314 ± 0.317	288 ± 7	15440 ± 98	13.6 ± 0.1
		150	85.920 ± 0.339	85.066 ± 0.364	62.496 ± 0.403	265 ± 44	16038 ± 178	14.1 ± 0.2
		200	85.876 ± 0.326	84.988 ± 0.378	62.435 ± 0.389	283 ± 26	15693 ± 51	13.8 ± 0.1
		250	19.158 ± 22.271	15.996 ± 23.038	11.712 ± 16.942	920 ± 226	15129 ± 1279	13.3 ± 1.1
		300	11.875 ± 0.606	8.383 ± 0.305	6.118 ± 0.232	1066 ± 63	16709 ± 323	14.7 ± 0.3
		350	11.716 ± 0.668	8.365 ± 0.272	6.098 ± 0.211	1002 ± 61	16873 ± 85	14.9 ± 0.1
		400	11.979 ± 0.514	8.461 ± 0.243	6.180 ± 0.173	544 ± 35	17519 ± 130	15.4 ± 0.1
		450	11.881 ± 0.413	8.434 ± 0.193	6.151 ± 0.138	464 ± 32	13282 ± 53	11.7 ± 0.1
		500	12.208 ± 0.557	8.537 ± 0.213	6.229 ± 0.172	550 ± 142	17954 ± 112	15.8 ± 0.1
	T2L	*50*	82.136 ± 0.245	85.281 ± 0.283	62.750 ± 0.294	177 ± 31	11730 ± 130	10.3 ± 0.1
		100	85.954 ± 0.314	85.092 ± 0.344	62.530 ± 0.376	310 ± 17	15405 ± 140	13.6 ± 0.1
		150	85.951 ± 0.393	85.070 ± 0.440	62.529 ± 0.467	168 ± 25	11899 ± 105	10.5 ± 0.1
		200	32.955 ± 34.594	30.637 ± 35.524	22.466 ± 26.107	835 ± 370	15872 ± 135	14.0 ± 0.1
		250	10.389 ± 0.418	7.314 ± 0.221	5.324 ± 0.159	990 ± 42	15988 ± 63	14.1 ± 0.1
		300	10.235 ± 0.405	7.258 ± 0.194	5.269 ± 0.148	795 ± 202	16669 ± 182	14.7 ± 0.2
		350	10.557 ± 0.614	7.322 ± 0.232	5.332 ± 0.174	462 ± 38	12217 ± 172	10.8 ± 0.2
		400	10.706 ± 0.667	7.371 ± 0.281	5.361 ± 0.207	505 ± 41	18004 ± 887	15.8 ± 0.8
		450	10.495 ± 0.342	7.408 ± 0.204	5.400 ± 0.152	458 ± 42	12554 ± 55	11.1 ± 0.1
		500	10.753 ± 0.498	7.449 ± 0.245	5.430 ± 0.183	517 ± 23	18641 ± 383	16.4 ± 0.3

Kernel	Dataset Variant	Feature Size	Result Metrics			Computation Time		
			RMSE$\pm std$	MAE$\pm std$	MAPE$\pm std$	Total Training$\pm std$ (s)	Total Prediction$\pm std$ (ms)	Prediction per Instance $\pm std$ (ms)
Matern	*T2L-CON*	*50*	7.964 ± 0.207	6.030 ± 0.165	4.330 ± 0.111	203 ± 2	15594 ± 158	13.7 ± 0.1
		100	8.013 ± 0.126	6.047 ± 0.120	4.345 ± 0.088	219 ± 170	14735 ± 165	13.0 ± 0.1
		150	7.861 ± 0.174	5.937 ± 0.121	4.253 ± 0.082	331 ± 165	15922 ± 137	14.0 ± 0.1
		200	7.848 ± 0.122	5.939 ± 0.083	4.269 ± 0.061	188 ± 2	15375 ± 116	13.5 ± 0.1
		250	7.953 ± 0.222	5.996 ± 0.173	4.305 ± 0.123	217 ± 28	16630 ± 183	14.6 ± 0.2
		300	7.887 ± 0.196	5.966 ± 0.149	4.284 ± 0.103	215 ± 1	15814 ± 117	13.9 ± 0.1
		350	8.024 ± 0.141	6.064 ± 0.126	4.370 ± 0.089	232 ± 2	16924 ± 94	14.9 ± 0.1
		400	7.927 ± 0.140	5.971 ± 0.116	4.294 ± 0.085	218 ± 1	13104 ± 80	11.5 ± 0.2
		450	7.941 ± 0.087	6.001 ± 0.094	4.327 ± 0.063	224 ± 1	13324 ± 59	11.7 ± 0.1
		500	8.016 ± 0.258	6.074 ± 0.175	4.363 ± 0.125	258 ± 4	17967 ± 118	15.8 ± 0.1
	T2L	*50*	7.147 ± 0.211	5.298 ± 0.140	3.804 ± 0.097	169 ± 105	11549 ± 62	10.2 ± 0.1
		100	7.110 ± 0.135	5.255 ± 0.117	3.768 ± 0.084	184 ± 1	15431 ± 71	13.6 ± 0.1
		150	7.230 ± 0.162	5.323 ± 0.104	3.812 ± 0.078	253 ± 111	15827 ± 146	13.9 ± 0.1
		200	7.228 ± 0.199	5.346 ± 0.144	3.837 ± 0.106	211 ± 1	15877 ± 89	14.0 ± 0.1
		250	7.088 ± 0.147	5.262 ± 0.121	3.784 ± 0.096	221 ± 1	16450 ± 244	14.5 ± 0.2
		300	7.114 ± 0.179	5.286 ± 0.122	3.805 ± 0.087	223 ± 1	16292 ± 75	14.3 ± 0.1
		350	7.050 ± 0.126	5.207 ± 0.092	3.738 ± 0.092	201 ± 1	12088 ± 43	10.6 ± 0.1
		400	7.128 ± 0.253	5.257 ± 0.166	3.778 ± 0.115	216 ± 1	13013 ± 91	11.5 ± 0.1
		450	7.120 ± 0.143	5.240 ± 0.115	3.772 ± 0.086	251 ± 4	17924 ± 626	15.8 ± 0.6
		500	7.022 ± 0.108	5.209 ± 0.091	3.740 ± 0.072	215 ± 6	12784 ± 56	11.3 ± 0.1
Linear	*T2L-CON*	*50*	8.596 ± 0.265	6.710 ± 0.223	4.797 ± 0.162	34 ± 1	15046 ± 100	13.2 ± 0.1
		100	8.478 ± 0.105	6.671 ± 0.085	4.783 ± 0.065	34 ± 1	15120 ± 95	13.3 ± 0.1
		150	8.541 ± 0.140	6.729 ± 0.102	4.805 ± 0.069	35 ± 1	15158 ± 58	13.3 ± 0.1
		200	8.461 ± 0.089	6.681 ± 0.077	4.804 ± 0.057	36 ± 1	15277 ± 65	13.4 ± 0.1
		250	8.615 ± 0.157	6.774 ± 0.111	4.852 ± 0.084	36 ± 1	15400 ± 88	13.6 ± 0.1
		300	8.727 ± 0.191	6.896 ± 0.127	4.929 ± 0.079	36 ± 1	15327 ± 115	13.5 ± 0.1
		350	8.778 ± 0.220	6.923 ± 0.195	4.964 ± 0.137	35 ± 1	15219 ± 64	13.4 ± 0.1
		400	8.726 ± 0.189	6.873 ± 0.168	4.929 ± 0.117	36 ± 1	15256 ± 43	13.4 ± 0.1
		450	8.804 ± 0.196	6.940 ± 0.148	4.967 ± 0.103	36 ± 1	15236 ± 67	13.4 ± 0.1
		500	8.789 ± 0.181	6.967 ± 0.151	4.996 ± 0.099	36 ± 1	15337 ± 39	13.5 ± 0.1
	T2L	*50*	7.297 ± 0.134	5.688 ± 0.092	4.078 ± 0.069	35 ± 1	14946 ± 44	13.2 ± 0.1
		100	7.367 ± 0.132	5.773 ± 0.102	4.131 ± 0.084	35 ± 1	15034 ± 138	13.2 ± 0.1
		150	7.390 ± 0.163	5.781 ± 0.106	4.148 ± 0.082	35 ± 1	15093 ± 74	13.3 ± 0.1
		200	7.441 ± 0.154	5.810 ± 0.121	4.160 ± 0.085	35 ± 1	15313 ± 124	13.5 ± 0.1
		250	7.533 ± 0.195	5.875 ± 0.133	4.198 ± 0.088	35 ± 1	15333 ± 106	13.5 ± 0.1
		300	7.481 ± 0.164	5.821 ± 0.139	4.187 ± 0.106	36 ± 1	15283 ± 99	13.5 ± 0.1
		350	7.501 ± 0.164	5.874 ± 0.118	4.210 ± 0.078	36 ± 1	15271 ± 45	13.4 ± 0.1
		400	7.521 ± 0.167	5.860 ± 0.157	4.208 ± 0.117	36 ± 1	15286 ± 89	13.5 ± 0.1
		450	7.636 ± 0.152	5.926 ± 0.135	4.249 ± 0.091	36 ± 1	15271 ± 55	13.4 ± 0.1
		500	7.637 ± 0.128	5.947 ± 0.112	4.269 ± 0.081	32 ± 1	11087 ± 57	9.8 ± 0.1

Kernel	Dataset Variant	Feature Size	Result Metrics			Computation Time		
			RMSE$\pm std$	MAE$\pm std$	MAPE$\pm std$	Total Training$\pm std$ (s)	Total Prediction$\pm std$ (ms)	Prediction per Instance $\pm std$ (ms)
Polynomial (Degree 2)	*T2L-CON*	*50*	9.052 ± 0.258	7.035 ± 0.161	5.071 ± 0.113	35 ± 1	14553 ± 116	12.8 ± 0.1
		100	43.983 ± 2.170	30.508 ± 0.683	22.253 ± 0.522	38 ± 1	15349 ± 32	13.5 ± 0.1
		150	36.305 ± 1.860	25.233 ± 0.646	18.401 ± 0.480	37 ± 1	14806 ± 105	13.0 ± 0.1
		200	30.082 ± 2.221	21.023 ± 0.750	15.296 ± 0.570	38 ± 1	15527 ± 269	13.7 ± 0.2
		250	28.299 ± 1.552	19.753 ± 0.715	14.415 ± 0.537	37 ± 1	14937 ± 100	13.1 ± 0.1
		300	26.754 ± 2.134	18.734 ± 0.702	13.629 ± 0.470	39 ± 1	15589 ± 117	13.7 ± 0.1
	T2L	*50*	7.895 ± 0.156	6.163 ± 0.108	4.445 ± 0.074	36 ± 1	15058 ± 32	13.3 ± 0.1
		100	38.226 ± 2.295	26.438 ± 1.030	19.272 ± 0.744	37 ± 1	15399 ± 116	13.6 ± 0.1
		150	30.335 ± 0.852	21.821 ± 0.525	15.884 ± 0.375	37 ± 1	15175 ± 39	13.4 ± 0.1
		200	25.270 ± 0.896	18.251 ± 0.403	13.309 ± 0.287	38 ± 1	15449 ± 180	13.6 ± 0.2
		250	23.369 ± 0.959	16.555 ± 0.505	12.056 ± 0.354	38 ± 1	15274 ± 56	13.4 ± 0.1
		300	22.198 ± 0.978	15.809 ± 0.418	11.531 ± 0.314	39 ± 1	15628 ± 240	13.8 ± 0.2
Polynomial (Degree 3)	*T2L-CON*	*50*	60.150 ± 2.459	38.564 ± 1.031	28.237 ± 0.815	38 ± 1	14762 ± 173	13.0 ± 0.2
		100	43.629 ± 1.977	27.388 ± 0.702	20.092 ± 0.559	40 ± 1	15396 ± 69	13.6 ± 0.1
		150	37.725 ± 6.173	20.482 ± 0.887	15.071 ± 0.681	55 ± 1	14840 ± 136	13.1 ± 0.1
		200	40.027 ± 2.626	23.298 ± 0.679	17.099 ± 0.522	60 ± 1	15815 ± 71	13.9 ± 0.1
		250	37.045 ± 3.412	20.401 ± 1.018	14.974 ± 0.763	99 ± 14	15238 ± 405	13.4 ± 0.4
		300	40.059 ± 4.734	20.364 ± 1.202	14.923 ± 0.935	321 ± 398	15857 ± 133	14.0 ± 0.1
	T2L	*50*	48.922 ± 2.765	32.846 ± 1.044	23.912 ± 0.716	39 ± 1	15147 ± 53	13.3 ± 0.1
		100	37.701 ± 1.997	24.616 ± 0.880	17.990 ± 0.632	40 ± 1	15593 ± 226	13.7 ± 0.2
		150	26.888 ± 1.755	17.147 ± 0.615	12.557 ± 0.467	57 ± 1	15326 ± 41	13.5 ± 0.1
		200	34.223 ± 1.490	20.509 ± 0.671	15.046 ± 0.540	59 ± 1	15602 ± 142	13.7 ± 0.1
		250	30.963 ± 3.219	17.363 ± 0.967	12.752 ± 0.730	95 ± 1	15504 ± 65	13.6 ± 0.1
		300	33.497 ± 4.686	17.865 ± 0.995	13.106 ± 0.995	117 ± 1	15783 ± 111	13.9 ± 0.1

APPENDIX B: **Code snippet for serializing the density functions by transforming the 3-dimensional tensor into a 2-dimensional matrix**

```
shape = feature_data.shape

serial_feature_data = np.zeros((shape[0],(shape[1] * shape[2])))

df_length = np.ma.size(feature_data,2)

for i in range(shape[0]):
  for j in range(shape[1]):
    for z in range(shape[2]):

      serial_feature_data[i,((j * df_length) + z)] = feature_data[i,j,z]
```