WAVELET SCATTERING AND GRAPH REPRESENTATIONS FOR ATOMIC STRUCTURES

By

Xavier Brumwell

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computational Mathematics, Science, and Engineering – Doctor of Philosophy

2021

ABSTRACT

WAVELET SCATTERING AND GRAPH REPRESENTATIONS FOR ATOMIC STRUCTURES

By

Xavier Brumwell

Machine learning for quantum chemistry has been gaining much traction in recent years. In this thesis, we address the problem of predicting the ground state energy from a collection of atoms defined by their positions and charges. The ground state energy of an atomic structure is invariant with respect to isometries and permutations. Additionally the energy is multiscale in nature and varies smoothly with movements of the atoms. We develop a wavelet scattering model which encodes all of these properties and scales better than commonly used computational chemistry models. We first demonstrate that this representation has excellent predictive ability on amorphous lithium silicon structures. We extend this model and improve its generalizability as displayed by predictions on several types of lithium silicon systems which are not included in the model training. Finally we take some of the principles from the wavelet scattering approach and apply them to a graph based model to generate a rich representation. This requires developing novel ways to encode the bond angle and multiscale aspects of the atomic structure for the graph. We test this model on a data set of quantum molecular dynamics simulations and get results that are competitive with the state of the art.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Matthew Hirn, for the guidance he has provided over the span of my graduate program. He has been extremely helpful in my transition into the world of research. He has also demonstrated patience and kindness that I hope to emulate toward others throughout my life.

I would also like to thank the members of my committee for giving their time. Drs. Alex Dickson, Mark Iwen, and Yue Qi have all been encouraging and helpful in their critiques and suggestions.

I am grateful for the collection of students and postdocs with whom I have been able to work over the last five years. Paul Sinz spent much time listening to my ideas and was always willing to help out. Jialin Liu, Kwang Jin Kim, and Michael Swift were gracious collaborators from materials science.

My wife, Emily, has been my greatest source of support. She has encouraged me through the challenges and celebrated with me in the victories. She made sacrifices so that I could succeed.

I am thankful that God has blessed me with this opportunity. To Him be the glory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Scientific discovery: Inputs, outputs, and supervised machine learning

Scientific inquiry is driven by learning the connection between inputs and outputs. For example: to generate a mathematical proof, one must discover the path from assumption (input) to conclusion (output) through the use of mathematical truths. Vaccines are created by leveraging scientific knowledge about how a chemical input will behave and selecting a particular input configuration such that the desired output (generation of antibodies) will be achieved.

The traditional approach to this discovery is through the derivation of closed form equations which match the observations of a process. Whether the calculations are direct or iterative, at the end there is some set of governing equations which describes with some accuracy the process. The paradigm of machine learning is a departure from this standard practice [1, 2]. In supervised machine learning, the purpose of the model design is to position oneself in reach of discovery. Rather than writing down a solution to the problem, one devises a structure that has certain flexibility, and that structure then will converge to the solution in ideal circumstances.

In a supervised machine learning setup, one has an input data space $\mathcal{X}$ and an output data space (labels) $\mathcal{Y}$. Let us assume that $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}$ which is sufficient to cover many cases in machine learning. One acquires a training set consisting of n_t pairs drawn from $\mathcal{X} \times \mathcal{Y}$:

$$\text{Training data} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} : 1 \leq i \leq n_t\} .$$

For a problem of interest in supervised machine learning, we assume that for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$, there is an unknown function $f : \mathcal{X} \to \mathcal{Y}$ which relates x to y.

$$y = f(x) , \quad \forall\, (x, y) \in \mathcal{X} \times \mathcal{Y} .$$

The goal in supervised machine learning is to infer the function f from the training data. In order to construct a machine learning model, one must make an educated guess as to the type of

function f. This means selecting a space of functions, $\mathcal{F}$, called the hypothesis space, to which the machine learning model will be constrained. Then, using the training data, one selects a function $f^\star \in \mathcal{F}$ that best approximates f according to some selected metric. One commonly used method is minimizing the mean squared error on the training set:

$$f^\star = \arg\min_{\widetilde{f} \in \mathcal{F}} \frac{1}{n_t} \sum_{i=1}^{n_t} |y_i - \widetilde{f}(x_i)|^2 \,. \tag{1.1}$$

There are several challenges with this approach. The function $f^\star$ may be elusive, and it may be incorrect. The first problem deals with optimization. While it is possible to derive closed-form solutions directly in some cases, e.g. ridge regression, the high dimensional hypothesis space, $\mathcal{F}$, of many problems in supervised machine learning requires one to iteratively optimize as a computationally tractable method of approximating $f^\star$. Generally, this process is not guaranteed to reach the best solution in the entire (global) function space, but will instead converge to a function that is only locally optimal. Much research is focused on developing optimization methods which converge to globally optimal functions with higher probability and speed [3, 4, 5].

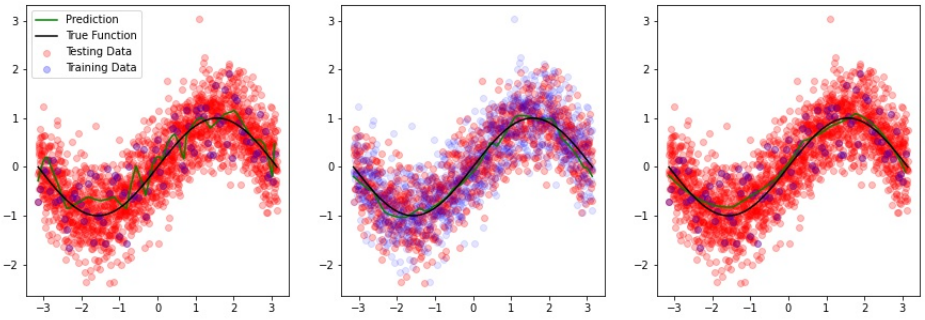

Figure 1.1: Models trained on data generated from the function $y = \sin(x) + \mathcal{N}(0, 0.25)$. On the left, the learned model is highly unstable due to overfitting. In the middle, the learned model is improved due to a higher number of training points. On the right, the learned model is improved by restricting the number of learnable parameters.

The second problem is addressed through regularization. When training on noisy data, especially if the data set is small and the machine learning model has many degrees of freedom, it is possible the model will adapt to the patterns in the noise. When this happens, the model is

"overfitted" to the training data, and is likely not the best predictor for the test data. Figure 1.1 shows an overfitted model and two solution pathways to avoid overfitting. Regularization imposes additional restrictions on the parameterization of the model. Unlike the design of the machine learning architecture, regularization does not block areas of the hypothesis space, but imposes penalties on them. This changes our minimization process to

$$f^\star = \arg\min_{\widetilde{f}\in\mathcal{F}} \frac{1}{n_t}\sum_{i=1}^{n_t} |y_i - \widetilde{f}(x_i)|^2 + \Omega(\widetilde{f}), \tag{1.2}$$

for some penalty function Ω. If f is parameterized by weights, a common selection is a penalty dependent on the magnitude of the weights. With smaller weights, or with fewer large weights, the model is more stable to perturbations in the data and will generally be a smoother function.

These problems can only be mitigated if we have selected a reasonable hypothesis space. If $f \notin \mathcal{F}$, then we have restricted the hypothesis space unsuitably for the problem, and there will be some irreducible error on the predictions. Even with the universal approximation theorems for neural networks [6, 7, 8, 9, 10], certain network designs may preclude the ability to approximate particular functions well. For this reason, one must take care in how the construction of the model affects the hypothesis space by making design choices specific to the problem at hand.

1.2 The Curse of Dimensionality

Different elements related to machine learning have improved over time. The reaches of computation have expanded. Optimization methods have become more refined. While we do address these areas, our main contribution is in designing hypothesis spaces $\mathcal{F}$ for machine learning algorithms. The appropriate hypothesis space can be vast in many machine learning problems. The size of the hypothesis space scales with the dimension of the inputs and makes finding the optimal model $f^\star$ improbable in most situations. This problem - the curse of dimensionality - manifests itself in two ways in machine learning. The optimization takes place in a high-dimensional, typically non-convex space, which means that convergence to the best solution is unlikely. Additionally, in the absence of a carefully constructed hypothesis space $\mathcal{F}$, one must have a number of training points n_t that grows exponentially in the number of input dimensions d to cover the input space.

Consider a Lipschitz function $f(x)$. To get a good approximation of this function, we need to sample well across its domain. If we consider a domain of $[0,1]^d$ and the ℓ_∞-norm as the distance measure, then to ensure that no point in the domain is no further than ϵ to at least one other point, we need to sample $n = O(\epsilon^{-d})$ points. Since f is Lipschitz, this also guarantees that the approximation error will be less than $C\epsilon$ where C is the Lipschitz constant of f. However, the scaling of the sampling requirement is too much for problems in high dimensions. It can be mitigated with increased smoothness of the underlying function, but typically one does not have control to freely adjust the target function in machine learning problems, and furthermore, to circumvent the curse of dimensionality the degree of smoothness would need to scale with dimension d, which is unrealistic.

In image processing [11], we can see a clear illustration of the problem of high dimensional data and also a solution. For a problem in image classification one has the space of images $\mathcal{X}$ and the class labels of the images $\mathcal{Y}$. The function $y = f(x)$ assigns a class label (e.g., cat, dog) to each input image x. Here, the input data, which is an image, is quite high-dimensional since each pixel is a separate dimension. For an $N \times N$ grayscale image, each data point is of dimension $d = N^2$. A color image is even larger at $d = 3N^2$ with standard practice being separation along red, green, and blue color channels. However, the data exists on a geometric structure - the grid - which can be leveraged to reduce the the dimension of the hypothesis space $\mathcal{F}$. By convolving an image with a filter, one assumes that the function which characterizes the interactions between neighboring pixels is uniform across the image. This approach significantly reduces the dimension of the functional space and has become the bedrock of modern image processing research.

Let us take an image $x : \mathbb{Z}^2 \to \mathbb{R}$ and a filter $h : \mathbb{Z}^2 \to \mathbb{R}$ with small support around the origin. Discrete convolution is defined as

$$x * h(u) = \sum_{v \in \mathbb{Z}^2} x(v)h(u - v), \quad u \in \mathbb{Z}^2 \tag{1.3}$$

With a translation $t \in \mathbb{Z}^2$ we define $x_t(u) = x(u - t)$. By convolving the filter with the translated image and defining $\tilde{v} = v - t$ we can show that the convolution operation is equivariant with respect

to translations.

$$
\begin{aligned}
(x_t * h)(u) &= \sum_{v \in \mathbb{Z}^2} x_t(v) h(u - v) \\
&= \sum_{v \in \mathbb{Z}^2} x(v - t) h(u - v) \\
&= \sum_{\tilde{v} \in \mathbb{Z}^2} x(\tilde{v}) h(u - t - \tilde{v}) \\
&= x * h(u - t) = (x * h)_t(u)
\end{aligned} \tag{1.4}
$$

By constraining the hypothesis space, it is more computationally tractable to obtain a good approximation $f^\star$ of the true label function f if the constraint is appropriate.

In this case, the hypothesis space has been constrained to the space of functions which are equivariant to translation. But to yield an image prediction, one needs an output which is invariant with respect to translations. For most image classification tasks, the label has no dependence on the position of the object of interest in the image. This means that the output of a neural network predicting this label should have no dependence on translations. To move in that direction, the set of convolved images are subsampled (commonly referred to as pooling), reducing the dimension of the image. Nonlinearities are also included to increase the size of the hypothesis space since convolution alone is a linear operator. Chaining convolutions against learned filters with subsampling layers and nonlinearities results in a convolutional neural network [12, 13, 14]. Since each constituent function is at a minimum equivariant to translations, the composition of these functions is also translation equivariant. Eventually, the subsampling will collapse the image to a single pixel. With no space dimension, the result is translation invariant and can be the input of a subsequent function which makes the image classification prediction.

There are two important things to note here that may not be immediately obvious. While translation invariance is what is desired, there is a substantial portion of the model that is translation equivariant. This is important because although global objects can move indiscriminately (a car on the left side vs the right side of an image), there is directional information of consequence that

is contained in the local scale (tires below vs above the car body). The second point is in regards to scales. Through the subsampling, we increase the effective size (receptive field) of the filters in each layer of convolutions. By the end of the network, the filters are acting on global information. However, it is not the information of every pairwise interaction of pixels, but the aggregates of the interactions from small neighborhoods. The CNN captures the information from a variety of scales, but restricts the hypothesis space to only operate on the interactions inside of neighborhoods for each scale. We see from this example how dramatically the hypothesis space that we select can constrain our model. This can be very helpful if the constraints are encoded so that the hypothesis space is restricted but not collapsed beyond what is necessary.

1.3 Machine Learning and Quantum Chemistry

In this thesis, we look at the application of supervised machine learning to quantum chemistry. This field has had explosive growth over the last several years yielding successful kernel-based [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25] as well as neural-network based models [26, 27, 28, 29, 30, 31, 32, 33, 34] for predicting physical properties of atomic structures. There are many things we know to be true about chemical systems which can be leveraged in the learning process. The input data in this case is the set of positions and charges for each atom in the system which we will define as

$$x = \{(Z_k, R_k) \in \mathbb{N} \times \mathbb{R}^3\}_{k=1}^{N} \tag{1.5}$$

for some atomic system x with N atoms. Our primary goal in this thesis will be to predict the ground state energy of the atomic system using machine learning methods.

1.3.1 Necessary Chemistry

We begin by considering the basis for this type of data and the properties with which it is imbued. The time-independent Schrödinger equation is a cornerstone of quantum chemistry [35], where H is the Hamiltonian operator and Ψ is the wave function for an atomic system with N_e electrons and

N nuclei.

$$H\Psi(r_1, r_2, ...r_{N_e}, R_1, R_2, ...R_N) = E\Psi(r_1, r_2, ...r_{N_e}, R_1, R_2, ...R_N) \tag{1.6}$$

Solving this eigenvalue problem efficiently and accurately for energy levels E is of intense interest to the chemical community. The Hamiltonian is

$$H = -\frac{1}{2}\sum_{i=1}^{N_e}\nabla_i^2 - \frac{1}{2}\sum_{A=1}^{N}\frac{1}{M_A}\nabla_A^2 - \sum_{i=1}^{N_e}\sum_{A=1}^{N}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N_e}\sum_{j>i}^{N_e}\frac{1}{r_{ij}} + \sum_{A=1}^{N}\sum_{B>A}^{N}\frac{Z_A Z_B}{R_{AB}} \tag{1.7}$$

where the terms are the kinetic and potential energy for the electrons (indexed by i and j) and nuclei (indexed by A and B).

The Born-Oppenheimer approximation constitutes a significant and commonly used reduction of the Hamiltonian [36]. It operates on the idea that since the scale difference between the electrons and nuclei is so significant, assuming fixed locations for the nuclei creates only minimal errors in the model. This approximation reduces the Hamiltonian to the electronic Hamiltonian.

$$H_{elec} = -\frac{1}{2}\sum_{i=1}^{N_e}\nabla_i^2 - \sum_{i=1}^{N_e}\sum_{A=1}^{N}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N_e}\sum_{j>i}^{N_e}\frac{1}{r_{ij}} \tag{1.8}$$

which has corresponding electronic wave function Ψ_{elec}. The second term from the Hamiltonian disappears because the nuclei kinetic energy is zero and the fifth term, (nucleus-nucleus repulsion) is reduced to a constant. We include the latter when considering the total energy which is defined as

$$E_{tot} = E_{elec} + \sum_{A=1}^{N}\sum_{B>A}^{N}\frac{Z_A Z_B}{R_{AB}} \tag{1.9}$$

To solve for the energy in the Schrödinger equation, one optimizes over the wave function Ψ. This is tractable for only the smallest atomic systems. There are a variety of methods which chemists use to compute molecular energies which offer a trade-off between high accuracy and low computational cost. Methods that generate properties with usable accuracy will typically scale as $O(N^3)$ at best. There are many applications in chemistry where high throughput is necessary to explore the chemical space sufficiently. There are also challenges in scaling to larger systems while maintaining accuracy and keeping computational costs low. For these reasons, there will always be a demand for better computational methods in quantum chemistry.

The variational method is one means of simplification that allows a faster approach to the solution. It begins with the principle that the energy resulting from any guessed wave function will necessarily be an upper bound on the true energy of the ground state. The set of wave functions is first constrained to those which are physically qualified in that they are continuous everywhere and quadratic integreble. At this point, it is still intractable to solve for the correct wave function so a subset must be chosen in some reasonable way. Letting E_1 denote the ground state energy,

$$E_1 = \inf_{\Psi \in \mathcal{H}} \langle \Psi, H\Psi \rangle \leq \inf_{\Psi \in \tilde{\mathcal{H}}} \langle \Psi, H\Psi \rangle, \quad \text{for} \quad \tilde{\mathcal{H}} \subset \mathcal{H} \tag{1.10}$$

where $\mathcal{H}$ is the Hilbert space of all possible wave functions and $\tilde{\mathcal{H}}$ is the subspace defined by the wave function method [37, 38]. It is not guaranteed that the optimal solution is contained in the chosen subset, but it can still be a good approximation. However, the optimization remains challenging because the wave function remains high dimensional. Due to this, it is only practical to perform this method on small molecules and short timescales.

Density functional theory is another common approach which can be used for much larger atomic structures [39, 40]. DFT replaces the problem of determining the wave function with the problem of designing a functional to operate on the electronic density. For some electronic density

$$n(u) = N_e \int dr_2 \cdots \int dr_{N_e} |\Psi(u, r_2 \cdots r_{N_e})|^2, \quad u \in \mathbb{R}^3 \tag{1.11}$$

one minimizes

$$E(n) = F(n) + \int V(u) n(u) du \tag{1.12}$$

for some pre-defined function F not depending on the system and

$$V(u) = \sum_{k=1}^{N} \frac{Z_k}{|u - R_k|} \tag{1.13}$$

Solving the minimization will yield the ground state energy. The accuracy of that energy depends upon how well F is defined. Early implementations of this approach lacked in accuracy, but more recent developments have allowed the method to be useful for a variety of chemical problems. DFT is not accurate enough for precise computations on small molecules, and it scales as $O(N^3)$ making it computationally infeasible for the largest structures. But in the middle regime DFT is a popular compromise to maintain good accuracy while computing properties on large structures.

1.3.2 Adapting Machine Learning to Chemistry

In order to efficiently create an effective machine learning representation in this context, one needs to encode information from chemistry. The total dimension of an input system is $4N$ with $R_k \in \mathbb{R}^3$ and $Z_k \in \mathbb{N}$. The size of the output space depends on the desired property. The energy of a molecule is a global feature which is then a single scalar per system. Predicting forces will yield an output space of $3N$ - one three-dimensional vector per atom. For the present, let us consider energy prediction. There are several properties of atomic systems that can be leveraged to reduce the hypothesis space. In particular, the energy of an atomic system is invariant with respect to global isometries. While pairwise distances between atoms matter, the locations of the atoms in space are arbitrary, and therefore translation information can be removed to collapse the input dimension. Similarly, reflections only imply a flip of the axis which has been placed upon the atomic system and does not stem from the system itself. Rotations have to be treated more carefully. Taking the angles of all atom triplets automatically puts us in $O(N^3)$ scaling. But while a system is invariant with respect to global rotations, local rotational information is important. It is true that all the angles of a system can be recovered from the complete pairwise distances, but it is unnecessary to place this burden on the learning algorithm when we know how to compute it. Aside from isometries, we have permutations. In a CNN, the filters and element-wise functions operate on each pixel in an identical manner. In an accurate representation of an atomic structure, each of the atoms should be treated identically regardless of their position or ordering index in the representation. The only allowable distinction is the atom type.

Scales are another aspect of the representation that must be considered. The nature of the interactions between pairs of atoms depends on the distance between them. A representation that captures this phenomena will have distinct portions dedicated to each length scale. This allows each of the portions to have unique generating functions that represent that scale well. However, there must be no discontinuity of the representation with respect to the atom positions. In particular, the representation needs to be smooth as the interaction between pairs of atoms move between scales.

With some base knowledge of how a representation should act the question then is what

representation should be used for atomic systems so that it simultaneously carries the mentioned invariants while maintaining the relevant information of the system. One could take $\sum_k Z_k$ as a set of features. This would be correctly invariant, but would be unable to distinguish between many dissimilar molecules since it contains no geometric information. On the other hand, one could take a sum over all the pairwise distances in the structure. This would also be appropriately invariant, but the immediate collapse of information creates a representation that maps many dissimilar atomic structures to the same point.

One of the earliest commonly cited works in the field of machine learning for quantum chemistry introduces the Behler-Parinello symmetry functions [26]. These functions encode the local environment of each atom in a system and are made up of a sum over all atoms of a Gaussian evaluated at the pairwise distance times a cutoff function,

$$G_i^1 = \sum_{j \neq i} \exp^{-\eta(R_{ij} - R_s)^2} f_c(R_{ij}) \tag{1.14}$$

where R_{ij} is the distance between atoms i and j, η and R_s are parameters, and f_c is a radial cutoff function. A similar 3-body term in constructed as

$$G_i^2 = 2^{1-\zeta} \sum_{j,k \neq i} (1 + \lambda \cos\theta_{ijk})^\zeta \exp^{-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \tag{1.15}$$

Each environment encoding G is passed through a neural network sharing parameters across atoms to get atom-wise energy which is then summed for the system energy prediction. This network is rudimentary but was significant in its advancement of the union of atomic systems with neural networks.

In 2012, Rupp et al [17] created the Coulomb matrix representation which was among the earliest works applying machine learning methods to atomic systems. The Coulomb matrix was constructed as a representation for a molecule with entries

$$M_{ij} = \left\{ \begin{array}{ccc} 0.5 Z_i^{2.4} & \forall & i = j \\ \frac{Z_i Z_j}{|R_i - R_j|} & \forall & i \neq j \end{array} \right\} \tag{1.16}$$

This is rotation and translation invariant since it only involves pairwise distances. However, this matrix is permutation equivariant. To remedy this, they diagonalized the matrix and padded

zeros for smaller molecules. The property predictions are then made as a learned weighted sum of Gaussians of the distances between the eigenvalues of the diagonalized matrices. While the Coulomb matrix contains some useful information, it is ridged and collapses the information down to N almost immediately. As a kernel-based method it is also very unlikely to predict well on molecules with distant representations.

Shortly after the introduction of Couloumb matrices, Bartok et al proposed a new kernel-based method called Smooth Overlap of Atomic Positions (SOAP) [18]. They examined various atomic neighborhood descriptors and concluded that the similarity measure between two representations is the most important facet of the predictive model. Their kernel is based on the inner product of atomic neighbor densities which are constructed using a spherical harmonic expansion of atom centered Gaussians. This inner product works out to be like rotating one local environment into the other, making the representation rotation invariant without a prohibitively expensive optimization over $SO(3)$, the group of 3D rotations. They demonstrate the faithfulness of their method by optimizing over the descriptor to recover target atomic geometries.

The next step that many researchers in this field took was the jump to a neural network driven approach. Kristof Schütt et al developed SchNet [41] which operates on the pairwise distances computed over a set of basis functions along with a learned embedding of the atomic charges. Operating on pairwise distances rather than positions removes all global translation and rotation information. The network has shared weights across the atoms so there is no dependency upon atom ordering. It has two types of modules which alternate: continuous-filter convolutions and interaction blocks. The convolution blocks combine information between atoms to generate a set of representations, and the interaction blocks combine the different representations to create rich features for each atom. This cycle is repeated until the output layer. The output is the sum of atom centered outputs which makes the algorithm permutation invariant. With many learn-able weights, this method scales in accuracy with the size of the training data, performing poorly for small training sets but yielding nice results when trained on a larger set. It also introduces the ability to train with forces. The forces offer many additional labels per data point, and are the

negative derivative of the energy with respect to each atom position. We discuss this point in more detail later.

Tensor-Field networks provide a nice advantage over the previous stated methods in using non-angularly symmetric functions to process the molecules [34]. They use spherical harmonics as a basis for their atom interaction functions to encode more complex interactions between the atoms and a special nonlinearity to maintain equivariance with respect to rotations. They use a similar framework to SchNet as the base model but apply it in a unique way by training to place a missing atom from a molecule.

While an atomic graph with every pairwise distance defined is unique modulo isometries, angular information is not guaranteed to be learned in a network. In addition to this, in many cases researchers use only local pairwise distances which do not completely define the geometric information of the systems. Recently, Klicpera et al [42] addressed this issue by including the angles between bonded atoms as explicit inputs in a representation dubbed DimeNet . This inclusion results in state-of-the-art performances in the small data training regime on several benchmark datasets.

1.4 Overview

Here are the basic components that make up our model and the questions that lead us to develop the representation as we do. We begin with a set of training data $\{(x_i, E_1(x_i))\}_{i=1}^{n_t}$ consisting of n_t training points where each $x_i = \{(Z_k^{(i)}, R_k^{(i)})\}_{k=1}^{N_{x_i}}$ is an atomic state with N_{x_i} atoms and $E_1(x_i)$ is its ground state energy. We create a vectorized representation $\Phi = (\phi_\gamma(x))_{\gamma\in\Gamma} \in \mathbb{R}^{|\Gamma|}$ such that $|\Gamma|$, the number of features in the representation, has no dependence on the number of atoms in the system N_x. Taking that representation, we wish to learn a model to predict the ground state energy as

$$\tilde{E_1}(x) = \sum_{\gamma\in\Gamma} w_\gamma \phi_\gamma(x) \tag{1.17}$$

To create this model, we have to decide on several design components. In Chapter 2 we construct the representation $\Phi(x)$ taking into consideration several physical properties we have described: invariance to isometries, continuity with respect to atom positions, multi-scale, and invariance to

permutations.

In Chapter 3 we present a method to solve for the weights w_γ as well as a data set on which we test our methods. In Chapter 4 we introduce several new types of atomic structure data. We adapt the model fitting to be able to extrapolate well to the new data.

In Chapter 5 we shift paradigms and consider a new representation vector that has internal learned components. We translate many ideas from the previous model and discuss the advantages and drawbacks. In Chapter 6, we augment this model and demonstrate its predictive ability on a benchmark data set.

1.5 Contributions

The choices that we make throughout this thesis have two primary motivations: speed and accuracy. Our approach have some advantages over existing methods in each of these categories.

There are many representations for molecules that scale linearly with the number of atoms. The key to achieving this scaling is typically to consider only local pairwise interactions. Our method does not quite match this scaling for molecules. However, the proposed method encapsulates all length scales of the atomic system and we do offer an advantage in the context of periodic structures. Our method scales with the number of atoms in the unit cell while including the effects of the boundary condition. There are few machine learning methods that consider how to represent periodic structures, and none that offer the boundary effects without additional computational cost.

Our representation is relatively unique in that we capture higher order interactions without combinatorial scaling or restricting to neighborhoods. Many methods capture some of the information from higher body terms, but we argue that this is diluted and not rich enough to properly capture the information. Recently, more methods have been including three-body terms to their representations [42, 41]. Our method includes an explicit consideration for higher body terms while maintaining good scaling.

Finally, our methods also allow for expression in the spaces between atoms. While an atomic structure is completely defined by the positions of the atoms, the space between the atoms is where

the electrons interact. Additionally, this allows us to represent long range interactions in a more geometrically flexible way.

CHAPTER 2

WAVELET SCATTERING

2.1 A Basis in Image Processing

We now define a representation suitable for atomic structures. Recall that, for this type of data, a data point can be characterized as $x = \{(Z_k, R_k) \in \mathbb{N} \times \mathbb{R}^3\}_{k=1}^N$. To retain complete geometric information, we encode the atomic structure as a 3D image. This places the set of atoms into a single object and automatically gives us permutation invariance as the atoms are placed without order. Without atom-based indexing, the inclusion of an atom for a particular feature will be entirely based on its position. This allows higher-order interactions without the poor computational scaling of explicitly combining the atoms into groups. This is a strength over many existing methods which rely on indexing and may operate on selected pairs or at best triplets of atoms. If we use electronic densities then developing a functional to operate on it will be similar to DFT. However, we select something that can be computed in $O(N)$ time instead of $O(N^3)$ time and does not rely on specific chemical methods. Initially based on some earlier works [43, 44], we used a sum of atom-centered Gaussians to create an image visually similar to an electronic density. After some analysis, we found that placing Diracs at atom positions reduces the image complexity and gives similar results

$$\rho_x(u) = \sum_{k=1}^{N} Z_k \delta(u - R_k) \tag{2.1}$$

Using a single raised point for each atom removes the need for any hyperparameters. We can allow greater expression if we replace Z_k with a function $c(Z_k)$. In particular, we can design a function which outputs 0 or 1 depending on the atom type generating images which include only select atom types. We refer to this set of functions $c(Z_k)$ as the channels.

Let

$$\rho_x^c(u) = \sum_{k=1}^{N} c(Z_k) \delta(u - R_k) \tag{2.2}$$

denote the Dirac density with channel c, and let C be the set of all channels. Set

$$\overrightarrow{\rho}_x = (\rho_x^c)_{c \in C} \tag{2.3}$$

so that $\overrightarrow{\rho}_x$ collects all channels of our Dirac representation of x. Recall that our final goal is to construct a representation $\Phi(x) \in \mathbb{R}^{|\Gamma|}$ that is invariant to translations, rotations, and reflection of the atomic coordinates $(R_k)_{k=1}^N$ and is also invariant to indexing permutations. Since the intermediate representation $x \to \overrightarrow{\rho}_x$ is invariant to index permutations, our goal moving forward is to construct a mapping that takes as input a 3D image and outputs a vectorized representation of the image that is invariant to translations, rotations, and reflections of the image, and whose dimension does not depend on the number of voxels in the image. Let $\rho : \mathbb{R}^3 \to \mathbb{R}$ be any 3D image, and let

$$\rho \to S(\rho) \in \mathbb{R}^d \tag{2.4}$$

be such a representation. We may then take our representation of x to be

$$x \to \Phi(x) = (S(\rho_x^c))_{c \in C} \in \mathbb{R}^{d|C|} \tag{2.5}$$

We now begin describing how to construct the mapping S.

If we follow modern image processing techniques, we would input the images into a convolutional neural network, constructed with convolution layers interspersed with pooling layers. There are several issues with translating this method directly to our approach. In images, the orientation is meaningful, whereas in molecules it is arbitrary. For example, in MNIST, a data set of handwritten digits 0-9, the orientation is the difference between a 6 and a 9 [45]. In atomic structures, we want to eliminate the effect that a rotation has on the model output. There are two primary options to generate a model that matches the rotational invariance that atomic structures contain. The first is data augmentation. In this case, one would include multiple rotated versions of each molecule into the training set. While this would not guarantee that the model maps rotated molecules to the same representation, it could encourage the model to make similar predictions for a single structure at its various rotations. However, this increases the training burden significantly, and it seems insensible to hope the model will be coerced to reflect something when we have the mechanics

to encode it ourselves. Additionally, since SO(3) is 3-dimensional, the number of additional data points will be considerably more than in the case for images. Even if a sufficient number of data points are included, the best that the representation could be is approximately rotation invariant. Since this is mismatched with the precise rotation invariance in physics, this would present an irreducible error in the representation. The second option is to put constraints on the filters and network architectures which guarantee that the resulting representation is invariant to rotations. One approach is to design an intermediate representation that is equivariant to rotations as this ensures that no orientation information is added into or lost from the representation. We can then derive an invariant representation from the equivariant one.

The discretization of the images adds a complexity to the pursuit of rotational invariance. Images and our pseudo-densities both immediately suffer some loss of information in the conversion from analog to discrete. When translating a continuous angularly symmetric filter to the discrete setting, the error with respect to rotation invariance is inversely related to the filter size. Consider a 3x3 angularly symmetric filter. It is only rotation equivariant when considering rotations in 90 degree increments. The error is significant for all other angles, however it can be reduced as the filter size grows. Using stacks of small filters, like in typical image processing, would induce large errors on the representation. For this reason, the filters that we will use will be defined on large grids, even if they have a small receptive field.

Assuming then that we are using filters defined at many points, a gradient descent type training process becomes intractable since we are in 3D. Defining these filters analytically gives us faster computation, and they do not require training to optimize. If we select filters which represent the systems well, then we only need machine learning to combine the representations.

2.2 The Design of Filters

For this work, we use wavelets as our filters. Wavelets are zero-average filters which are localized in both space and frequency [46]. Unlike in a learned convolutional network, here we can directly control what types of information the different filters are encoding. The wavelets we

use must be defined in 3D and we construct them in two pieces: a radial function and a spherical harmonic.

$$\psi^m_{\gamma\ell}(u) = Q_\gamma(r)Y^m_\ell(\phi,\theta), \quad u = (r,\phi,\theta) \in \mathbb{R}^3 \tag{2.6}$$

Since we are intending to obtain rotational equivariance with these filters, it is most natural to work in spherical coordinates with radius r, azimuthal angle $\phi \in [0, 2\pi]$, and polar angle $\theta \in [0, \pi]$. We select the spherical harmonics for specific angular properties, but the radial portion Q_γ is free to define. The spherical harmonic functions are indexed by parameters $\ell = 0, 1, 2, \ldots$ and $-\ell \leq m \leq \ell$. After selecting a set of wavelets, we dilate the base wavelets to generate a set of wavelets that operate at a variety of scales.

$$\psi^m_{\gamma\ell j}(u) = 2^{-3j}\psi^m_{\gamma\ell}(2^{-j}u) \tag{2.7}$$

We compute a wavelet transform as

$$W\rho = \{\rho * \psi^m_{\gamma\ell j}(u) : \gamma \in \Gamma, \ell \geq 0, |m| \leq \ell, j \in \mathbb{Z}, u \in \mathbb{R}^3\} \tag{2.8}$$

Notice that the individual wavelet coefficient maps are defined using convolution.

$$(\rho * \psi^m_{\gamma\ell j})(u) = \int \rho(t)\psi^m_{\gamma\ell j}(u-t)dt \tag{2.9}$$

By using convolutions, we get an intermediate representation which is equivariant with respect to translations. The wavelet transform of the delta pseudo densities results in a set of 3D images with the corresponding wavelet emitted at each atom location and scaled by the corresponding $c(Z_k)$ coefficient.

$$(\rho^c_x * \psi^m_{\gamma\ell j})(u) = \sum_{k=1}^{N} c(Z_k)\psi^m_{\gamma\ell j}(u - R_k) \tag{2.10}$$

The interference patterns in the wavelet coefficients encode geometric properties of the atomic structure x at various scales and angular frequencies. With the wavelet transform we can choose the level of granularity for each of these parameters including many scales and a variety of angular frequencies for a richer representation or only including a minimal set for fast computation.

When convolving an image with angularly symmetric filters, local angular information is immediately discarded. While the rotational invariance is appropriate globally, it is too restrictive when working with local atomic environments at the beginning of the model. We build our wavelets with the spherical harmonics so that they are not angularly symmetric, but preserve rotation information in a specific way. The wavelet transform with our wavelets is not equivariant to rotations until we apply a special modulus [43, 44] to it which we define as

$$\sigma(\rho * \psi_{\gamma\ell j})(u) = \left[\sum_{m=-\ell}^{\ell} |\rho * \psi^m_{\gamma\ell j}(u)|^2 \right]^{1/2} \tag{2.11}$$

Using the spherical harmonics with the special modulus encodes local angular information while generating a representation that is equivariant to global rotations. We now prove this equivariance.

Theorem 1. *The special modulus of the wavelet scattering transform using our wavelets is equivariant with respect to rotations.*

$$\sigma(L_R\rho * \psi_{\gamma\ell j}) = L_R\sigma(\rho * \psi_{\gamma\ell j})$$

where $L_R\rho(u) := \rho(R^{-1}u)$ *for* $R \in SO(3)$.

Proof. For the set of spherical harmonics Y^m_ℓ with $\ell \geq 0$, $-\ell \leq m \leq \ell$ we have

$$Y^m_\ell : S^2 \to \mathbb{C} \tag{2.12}$$

We begin with a density $\rho : \mathbb{R}^3 \to \mathbb{R}$. We have the related Wigner D-matrices $D_\ell(R)$ with size $(2\ell + 1)\text{x}(2\ell + 1)$. They are related to the spherical harmonics by

$$L_R Y^m_\ell = \sum_{m'=-\ell}^{\ell} D_\ell(R)_{m,m'} Y^{m'}_\ell \tag{2.13}$$

Let us define $Y_\ell : S^2 \to \mathbb{C}^{2\ell+1}$ collapsing the m parameter as

$$Y_\ell(\phi, \theta) := \left(Y^m_\ell(\phi, \theta) \right)^{\ell}_{m=-\ell} \tag{2.14}$$

Then by rewriting Equation (2.13) we have

$$L_R Y_\ell = D_\ell(R) Y_\ell \tag{2.15}$$

Note that $D_\ell(R)$ is a unitary matrix. With a wavelet defined as

$$\psi_\ell^m(u) = Q(|u|)Y_\ell^m\left(\frac{u}{|u|}\right) \tag{2.16}$$

we consider a wavelet transform with the rotated image

$$L_R\rho * \psi_\ell^m(u) = \int_{\mathbb{R}^3} \rho(R^{-1}y)\psi_\ell^m(u-y)dy$$

Let $x = R^{-1}y, \quad dx = dy$

$$\begin{aligned}
&= \int_{\mathbb{R}^3} \rho(x)\psi_\ell^m(u-Rx)dx \\
&= \int_{\mathbb{R}^3} \rho(x)Q(|u-Rx|)Y_\ell^m\left(\frac{u-Rx}{|u-Rx|}\right)dx \\
&= \int_{\mathbb{R}^3} \rho(x)Q(|R^{-1}u-x|)Y_\ell^m\left(\frac{R(R^{-1}u-x)}{|R^{-1}u-x|}\right)
\end{aligned}$$

Consider then the special modulus

$$\begin{aligned}
|\sigma(L_R\rho * \psi_\ell)(u)|^2 &= \sum_{m=-\ell}^{\ell}\left[\int_{\mathbb{R}^3} \rho(x)Q(|R^{-1}u-x|)Y_\ell^m\left(\frac{R(R^{-1}u-x)}{|R^{-1}u-x|}\right)dx\right]^2 \\
&= \sum_{m=-\ell}^{\ell}\left[\int_{\mathbb{R}^3} \rho(x)Q(|R^{-1}u-x|)Y_\ell^m\left(\frac{R(R^{-1}u-x)}{|R^{-1}u-x|}\right)dx\right] \\
&\quad \cdot \left[\int_{\mathbb{R}^3} \rho(y)Q(|R^{-1}u-y|)\overline{Y_\ell^m\left(\frac{R(R^{-1}u-y)}{|R^{-1}u-y|}\right)}dy\right] \\
&= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3} \rho(x)\rho(y)Q(|R^{-1}u-x|)Q(|R^{-1}u-y|) \\
&\quad \cdot \sum_{m=-\ell}^{\ell} Y_\ell^m\left(\frac{R(R^{-1}u-x)}{|R^{-1}u-x|}\right)\overline{Y_\ell^m\left(\frac{R(R^{-1}u-y)}{|R^{-1}u-y|}\right)}dxdy
\end{aligned}$$

Further examining the inner summation we have

$$\begin{aligned}
\sum_{m=-\ell}^{\ell} Y_\ell^m\left(\frac{R(R^{-1}u-x)}{|R^{-1}u-x|}\right)\overline{Y_\ell^m\left(\frac{R(R^{-1}u-y)}{|R^{-1}u-y|}\right)} &= \\
&= \sum_{m=\ell}^{\ell} L_{R^{-1}} Y_\ell^m\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right) L_{R^{-1}}\overline{Y_\ell^m\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)} \\
&= \sum_{m=\ell}^{\ell}\left(\sum_{m'=-\ell}^{\ell} D_\ell(R^{-1})_{m,m'} Y_\ell^{m'}\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right)\right) \\
&\quad \left(\sum_{m''=-\ell}^{\ell}\overline{D_\ell(R^{-1})_{m,m''}}\,\overline{Y_\ell^{m''}\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)}\right) \\
&= \sum_{m=\ell}^{\ell}\left[D_\ell(R^{-1})Y_\ell\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right)\right]_m\left[\overline{D_\ell(R^{-1})}\,\overline{Y_\ell\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)}\right]_m \\
&= \left\langle D_\ell(R^{-1})Y_\ell\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right), D_\ell(R^{-1})Y_\ell\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)\right\rangle \\
&= \left\langle D_\ell(R^{-1})^* D_\ell(R^{-1})Y_\ell\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right), Y_\ell\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)\right\rangle \\
&= \left\langle \mathrm{I}\cdot Y_\ell\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right), Y_\ell\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)\right\rangle \\
&= \sum_{m=-\ell}^{\ell} Y_\ell^m\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right) Y_\ell^m\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)
\end{aligned}$$

Placing this back in the original equation, we get

$$\begin{aligned}
|\sigma(L_R\rho * \psi_\ell)(u)|^2 &= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3} \rho(x)\rho(y)Q(|R^{-1}u-x|)Q(|R^{-1}u-y|) \\
&\qquad \cdot \sum_{m=-\ell}^{\ell} Y_\ell^m\left(\frac{(R^{-1}u-x)}{|R^{-1}u-x|}\right)\overline{Y_\ell^m\left(\frac{(R^{-1}u-y)}{|R^{-1}u-y|}\right)}dxdy \\
&= \sum_{m=-\ell}^{\ell}\left[\int_{\mathbb{R}^3}\rho(x)Q(|R^{-1}u-x|)Y_\ell^m\left(\frac{R(R^{-1}u-x)}{|R^{-1}u-x|}\right)dx\right]^2 \\
&= \sum_{m=-\ell}^{\ell}|\rho * \psi_\ell^m(R^{-1}(u))|^2 = |L_R\sigma(\rho * \psi_\ell)(u)|^2
\end{aligned}$$

□

With the rotation equivariance demonstrated, we turn to the radial portion of our filters. The

wavelets we use have radial component

$$Q_{n\ell}(r) = K_{n\ell}(\beta)|u|^{\ell} L_{n-\ell-1}^{\ell+1/2}(|u|^2/(2\beta^2))e^{-|u|^2/(2\beta^2)} \tag{2.17}$$

where $K_{n\ell}$ is a normalizing factor that depends on β. We set β as a hyperparameter controlling the width of the wavelets. L_k^v is an associated Laguerre polynomial. We use the Laguerre polynomials so that our wavelets can be defined simply in frequency, a point which we discuss later. The constant $K_{n\ell}(\beta)$ is defined as

$$K_{n\ell}(\beta) = \frac{2^{n-\ell-1}(n-\ell-1)!}{(\sqrt{2\pi})^3 \beta^{2n+1}} \sqrt{\frac{4\pi}{2\ell+1}} \tag{2.18}$$

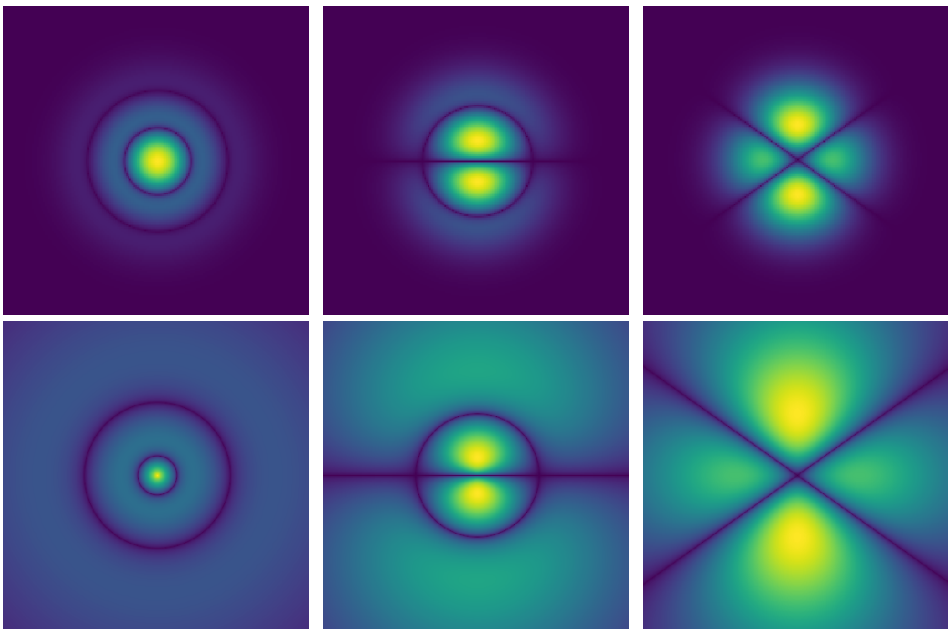

Figure 2.1: Top row left to right: Density plot cross sections in the xz-plane of atomic orbital wavelets for $(n, \ell, m) = (3, 0, 0)$, $(3, 1, 0)$, and $(3, 2, 0)$. Bottom Row: Corresponding plots for the hydrogen atom orbitals ($3s$, $3p$, and $3d$ orbitals, respectively).

Our wavelets are then defined as

$$\psi_{n\ell}^m(u) = Q_{n\ell}(r) Y_\ell^m(\phi, \theta), \qquad n \geq 1, \quad 0 \leq \ell < n, \quad |m| \leq \ell \tag{2.19}$$

The wavelets are designed to imitate the hydrogen orbitals. Indeed, $(n, \ell) = (1, 0), (2, 0)$, and $(2, 1)$ correspond to the $1s$, $2s$, and $2p$ orbitals, respectively, with similar correspondences for larger values of n. While the hydrogen atomic orbitals have exponential scaling, here, we

use a Gaussian function, which mimics the well-known Gaussian type orbitals from the quantum chemistry literature [47, 48], designing our wavelet filters to be more localized decaying at e^{-r^2} as opposed to the e^{-r} radial decay of the hydrogen orbitals as seen in Figure 2.1.

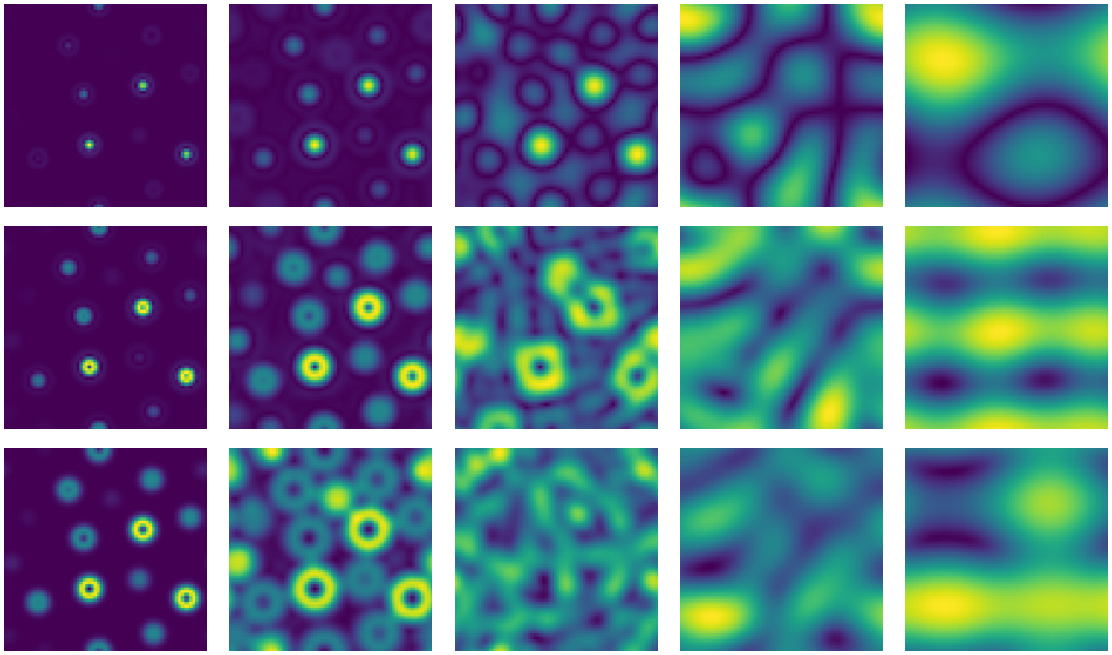

Figure 2.2: Cross-sections of the first order nonlinear, equivariant maps $\sigma(\rho_x * \psi_{n\ell j})(u)$. The $\log_2$ scales $j = 0, 1, 2, 3, 4$ increase from left to right, respectively, and the angular quantum number $\ell = 0, 1, 2$ from top to bottom, respectively, with $n = 3$.

2.3 Combining Scales

With a Fourier series, one has to compute responses over a large set of frequencies to get a rich representation of an image. Similarly, we expand our wavelet basis to encode information at a variety of scales by dilating the mother wavelet at various scales. The expansion of the filter set to operate across scales means that we will have distinct representations for each scale. Small filters will encode the interactions within atom neighborhoods, while large filters will encode the distribution of atoms across the entire atomic structure as can be seen in Figure 2.2.

While separating the information from the atomic systems into different scales is useful, re-combining these scales gives features even greater predictive ability. Since the special modulus is

equivariant to isometries, composing it with itself is also equivariant.

$$\sigma(\sigma(\rho * \psi_{n_1\ell_1 j_1}) * \psi_{n_2\ell_2 j_2}) = \left(\sum_{m=-\ell_2}^{\ell} |\sigma(\rho * \psi_{n_1\ell_1 j_1}) * \psi^m_{n_2\ell_2 j_2}(u)|^2 \right)^{1/2} \tag{2.20}$$

Proposition 1.1. *Let $f, g : L^1(\mathbb{R}^3) \to L^1(\mathbb{R}^3)$ be rotation equivariant maps. Then $f \circ g$ is also rotation equivariant.*

Proof. Since f is rotation equivariant we have

$$f(L_R\rho) = L_R f(\rho) \tag{2.21}$$

and similarly for g. Therefore

$$\begin{aligned}(f \circ g)(L_R\rho) &= f(g(L_R\rho)) \\ &= f(L_R g(\rho)) \\ &= L_R f(g(\rho)) \\ &= L_R(f \circ g)(\rho)\end{aligned} \tag{2.22}$$

□

Each subsequent layer will act on the 3D image from the previous layer instead of acting on the input dirac pseudo-density. In Figure 2.3, we see that the coupling of scales gives us different, more complex interference patterns between the atoms compared to the first layer wavelet coefficients. The processing up to this point is quite similar to that of a CNN. We applied a filter, then a nonlinearity, then another filter. However, there are several notable differences. In a CNN, one learns the filters whereas we have designed filters to have specific properties. Our nonlinearity is more complex than a standard CNN's ReLU or sigmoid since we require the special modulus to get rotation equivariance. We do not pool, because we want precise control of the resolution, and the multiscale nature of our filters allows us to capture the various scales without pooling.

Here we apply some of the convention of image processing and require subsequent filters to have a larger or equal scale of application compared to earlier layers. That means that we select

only combinations where $j_2 \geq j_1$. A smaller wavelet in the first layer will aggregate information between local atoms into neighborhoods. The larger second layer wavelet can then combine the information across the different neighborhoods to get information that is representative of the whole structure and has been built up hierarchically. Applying a large filter in the first layer collapses information from across large portions of the image, and in doing this, they effectively diminish the resolution of local neighborhoods. This is helpful to get a coarser representation of the entire structure, but it would not be helpful to apply a smaller wavelet in the second layer because the small scale interactions that the wavelet would capture have been destroyed.

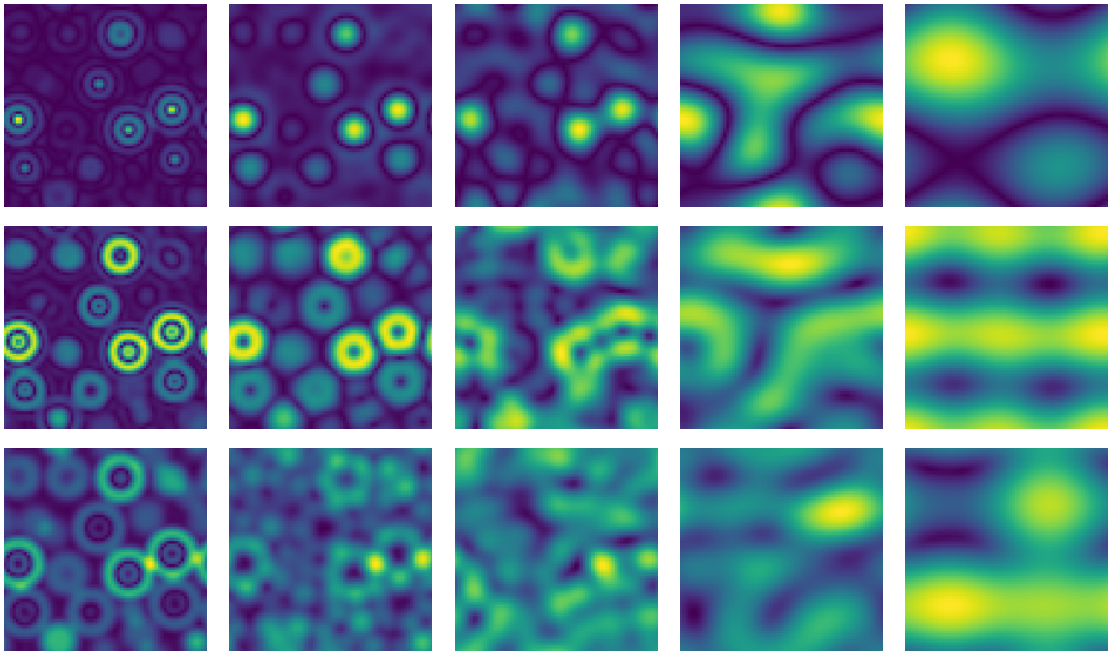

Figure 2.3: Cross-sections of the second order nonlinear, equivariant maps $\sigma(\sigma(\rho_x * \psi_{n_1\ell_1 j_1}) * \psi_{n_2\ell_2 j_2})(u)$ for $(n_1\ell_1 j_1) = (1, 3, 1)$, which is the second from the top and second from the left in Figure 2.2. The $\log_2$ scale, j_2, and ℓ_2 vary the same as in Figure 2.2, and $n_2 = 3$.

Prior to making predictions on the chemical properties, we need to convert the equivariant feature maps we have built into invariant features. By integrating over the wavelet special modulus coefficients, we collapse location information within the system and turn all equivariance to

invariance,

$$\|\sigma(\rho * \psi_{n\ell j})\|_q^q = \int_{\mathbb{R}^3} |\sigma(\rho * \psi_{n\ell j})(u)|^q du \tag{2.23}$$

$$\|\sigma(\sigma(\rho * \psi_{\lambda_1}) * \psi_{\lambda_2})\|_q^q = \int_{\mathbb{R}^3} |\sigma(\sigma(\rho * \psi_{\lambda_1}) * \psi_{\lambda_2})(u)|^q du \tag{2.24}$$

where $\lambda_1 = (n_1, \ell_1, j_1)$ and $q \in \{1, \frac{4}{3}, \frac{5}{3}, 2\}$. We refer to these as wavelet scattering coefficients after [49]. The selection of the powers q is motivated by the Thomas–Fermi–Dirac–von Weizsacker model in quantum chemistry, in which the $4/3$ scaling is used to approximate the exchange energy, the $5/3$ scaling is used to approximate the kinetic energy, and the power of 2 encodes an additional part of the kinetic energy and pairwise Coulombic interactions. The power $q = 1$ is also used since these integrals scale linearly with $\sum_k Z_k$. These features are now invariant with respect to the isometries present in chemical structures.

Proposition 1.2. *Let $f : L^1(\mathbb{R}^3) \to L^1(\mathbb{R}^3)$ be equivariant with respect to translations and rotations. Then*

$$\phi(\rho) = \int_{\mathbb{R}^3} f(\rho)(u) du, \quad \rho \in L^1(\mathbb{R}^3) \tag{2.25}$$

is invariant to translations and rotations.

Proof. Let $L_t\rho(u) := \rho(u - t)$ be the translation operator. Since f is equivariant to translations we have

$$f(L_t\rho) = L_t f(\rho) \tag{2.26}$$

Therefore

$$\begin{aligned} \phi(L_t\rho) &= \int_{\mathbb{R}^3} f(L_t\rho)(u) du \\ &= \int_{\mathbb{R}^3} L_t f(\rho)(u) du \\ &= \int_{\mathbb{R}^3} f(\rho)(u - t) du \end{aligned} \tag{2.27}$$

Let $v = u - t$ and $dv = du$. Then

$$\begin{aligned} \phi(L_t\rho) &= \int_{\mathbb{R}^3} f(\rho)(v) dv \\ &= \phi(\rho) \end{aligned} \tag{2.28}$$

Similarly for rotations we have

$$\begin{aligned}\phi(L_R\rho) &= \int_{\mathbb{R}^3} f(L_R\rho)(u)du \\ &= \int_{\mathbb{R}^3} L_R f(\rho)(u)du \\ &= \int_{\mathbb{R}^3} f(\rho)(R^{-1}u)du \end{aligned} \tag{2.29}$$

Now let $v = R^{-1}u$ and $dv = du$. Then

$$\begin{aligned}\phi(L_R\rho) &= \int_{\mathbb{R}^3} f(\rho)(v)dv \\ &= \phi(\rho) \end{aligned} \tag{2.30}$$

□

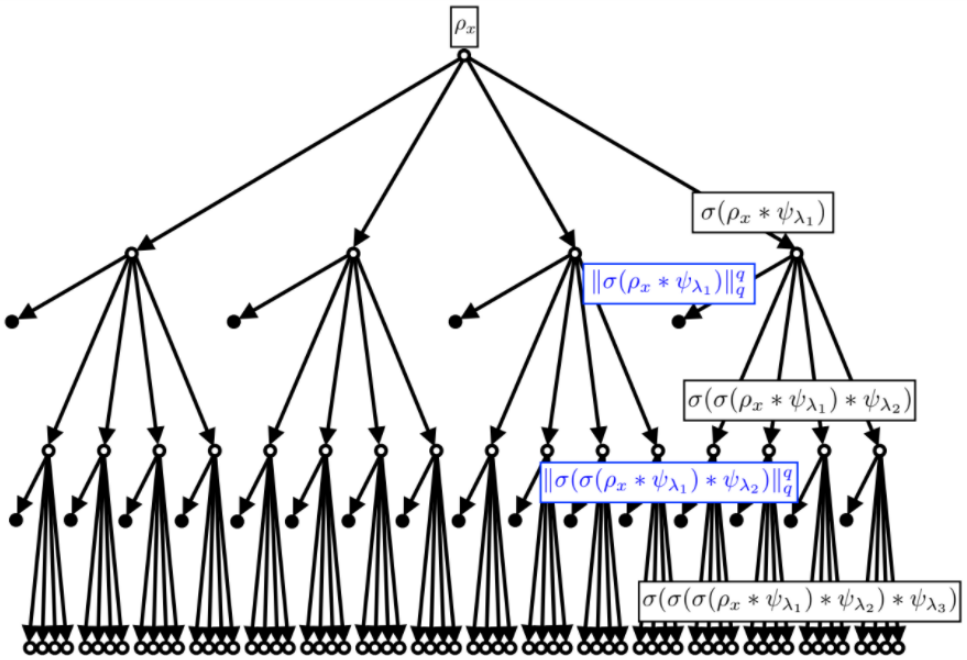

Figure 2.4: This tree diagram displays the general structure of a wavelet scattering representation. Each empty node is a 3D representation. Each filled in node is a feature of the representation. Each split represents a choice among the wavelets in the dictionary. Note that we can extract features from representations at any of the layers, and we can extend this structure as far down as computation allows.

As can be seen in Figure 2.4, we do not lose the first order features through the scattering process. All of the first order scattering coefficients are integrated and output from the algorithm

and then concatenated with the integrated second order scattering coefficients. We also add in "zero-order" features that have no geometric information and only add up the charge information. All together the scattering map S is defined as

$$\begin{aligned} S(\rho) := \{&\|\rho\|_q^q, \\ &\|\sigma(\rho * \psi_{\lambda_1})\|_q^q, \\ &\|\sigma(\sigma(\rho * \psi_{\lambda_1}) * \psi_{\lambda_2})\|_q^q\}_{\lambda_1,\lambda_2,q} \end{aligned} \tag{2.31}$$

where we recall $\lambda = (n, l, j)$. Using S and the intermediate representation $\overrightarrow{\rho}_x$ of the atomic state x, we define the representation Φ as

$$\Phi(x) = (S(\rho_x^c))_{c \in C} \tag{2.32}$$

This gives us a set of features with varying natures of interaction between the atoms. The framework allows us to continue adding layers as much as is computationally feasible. Each subsequent layer builds upon the previous layer in the same way. For the features that we compute, the additional computation to go beyond two layers does not seem to have sufficient added value to the representation to warrant inclusion.

2.4 Setting Parameters

With a theoretical basis of what we want to do, we now turn to the actual methods of implementation. Recall

$$\rho_x * \psi_{n\ell j}^m(u) = \sum_k Z_k \psi_{n\ell j}^m(u - R_k) \tag{2.33}$$

Let us assume $0 \leq j \leq J$. We control the size of the base (smallest) $j = 0$ wavelet so that there will be some interaction between the closest atoms. See Figure 2.5. Setting a smaller size would result in a feature representation that contained no overlap between atoms and generated no interference patterns, which would mean that the additional computation of the first order feature would provide no more information than a zero order feature. Setting the base too large could remove the option of learning small scale interactions. We compute the minimum distance between pairs of atoms

across the data set and denote it as Δ. To examine the smallest interaction, we take the wavelet with minimal radial support as the base, setting $n = 1$, $\ell = 0$. This cancels several terms from our wavelet definition.

$$\psi^0_{1,0}(u) = \exp\left(-\frac{|u|^2}{2\beta^2}\right) \tag{2.34}$$

We then set $\epsilon_o << 1$ as the minimum overlap. We select β to ensure

$$\psi^0_{1,0}\left(\frac{\Delta}{2}\right) = \exp\left(-\frac{(\Delta/2)^2}{2\beta^2}\right) \geq \epsilon_o \tag{2.35}$$

This solves to

$$\beta \geq \sqrt{\frac{(\Delta/2)^2}{2\ln(1/\epsilon_o)}} \tag{2.36}$$

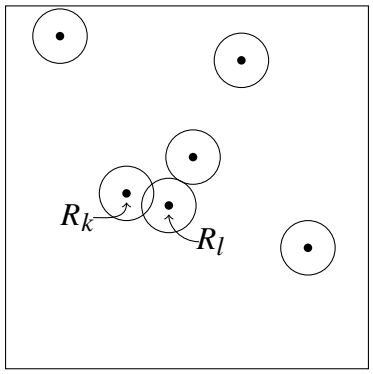

Figure 2.5: Molecule in state $x = \{(z_k, R_k)\}_{k=1,6}$. The neighborhoods of each R_i represent where $\psi^0_{1,0}(u - R_i) \geq \epsilon_o$. For this molecule, we would set the minimum width large enough so that the disc surrounding R_k intersects the point R_l.

We then set our maximum scale $j = J$ with a similar argument. We need to assess what maximum scale of interaction is appropriate for each type of molecular data that we consider. For the materials structures that are the primary consideration for this work, the maximum interaction of consequence is approximately the same length as the side of the bounding box of the material. If we use the same wavelet as before to ensure that the maximum scale, J, of every wavelet covers the desired width, we have

$$\psi^0_{J,1,0}(u) = 2^{-3J} \exp\left(-\frac{|2^{-J}u|^2}{2\beta^2}\right) \tag{2.37}$$

Since this is a Gaussian, we can ensure a contribution to a feature from distant atoms by controlling the standard deviation. In particular, we set 2 standard deviations to be equal to $B_e = (boxsize)/2$.

$$B_e = 2^J \sqrt{2}\beta \tag{2.38}$$

$$J = \log\left(\frac{B_e}{2\sqrt{2}\beta}\right) \tag{2.39}$$

We typically get a value $3 \leq J \leq 5$ for structures we consider. These parameters fall into the category of feature engineering.

The final parameter that we calculate to set up the wavelet transform computation is the sampling rate. This parameter is important because it dictates the aliasing error. Unlike a poor choice for J or β, a sampling rate that is too low will result in corrupt data. Rather than feature engineering, this decision controls the error on the representation. We set $\epsilon_a << 1$ smaller than than the other bounds. To avoid aliasing we need the essential support of $\widehat{\psi}^m_{n\ell}(\omega)$ to be contained in the box $[-\frac{\pi}{s}, \frac{\pi}{s}]^3$ where s is the sampling rate. Rather than restricting the frequency support of the wavelet, which is determined by β, we address this by expanding the size of the frequency domain that we compute. We need to solve for s which we do with a numerical optimizer.

$$\hat{\psi}^0_{n,0}(\frac{\pi}{s}, \xi, \zeta) = \left(\frac{\pi}{s}\right)^{2n-2} \exp\left(-\frac{\beta^2 \frac{\pi}{s}^2}{2}\right) \sqrt{4\pi} \leq \epsilon_a \tag{2.40}$$

The maximum frequency depends on the wavelet width β and, to a lesser degree, on the (n, ℓ) parameters. A wider support in frequency is equivalent to denser sampling in space. We begin by sampling at every Angstrom within the box, but increase the density to get the necessary resolution to be able to represent the system with low aliasing errors. Our sampling on the chemical system in space will be on the grid $\mathcal{G}_x = \{sa : a \in \mathbb{Z}^3 : \|a\|_\infty \leq B_e\}$ with L_x grid points in length. In frequency, evaluations are restricted to a grid $\Omega_x \subset [-\frac{\pi}{s}, \frac{\pi}{s}]^3$, with again L_x grid points along each edge. We balance the selection of n so that we have diverse features, but the computational cost is not too high.

For a typical set of systems that we process, we may set the overlap epsilon at $\epsilon_o = 0.1$ and the aliasing epsilon at $\epsilon_a = 10^{-6}$. With $n = \{1, 2, 3\}$, we get $\beta = 1.8$, $J = 4$, and $s = 0.1$. The

average width of the unit cell of the structures is about 10 Angstroms. From this, we get a cube of $104 \times 104 \times 104$ on which all the computations take place. Each voxel will correspond to a space in the structure of volume 10^{-3} cubic Angstroms. To approximate the integral that collapses the representations to isometry invariant features, we take a left-hand Riemann sum over the sampled points in the box. In total we will end up with around 3,700 features per structure including zero, first, and second order as well as the five density channels that we will introduce later.

2.5 Fast Frequency Calculations

We perform significant portions of the computation for our representation in frequency to reduce computational costs. Since we are working with mainly periodic materials, we need the boundary conditions of the convolutions to be circular so that atoms on opposite sides of the box appear next to each other. Due to the multiscale sizes of the wavelet filters $\psi^m_{n\ell j}$ a direct computation of the periodic convolution over the grid $\mathcal{G}_x$ will require $O(L_x^6)$ floating point operations. This computational cost can be significantly reduced by carrying out these computations in frequency space.

The Fourier transform of $\rho^c_x * \psi^m_{n\ell j}$ is:

$$\mathcal{F}[\rho^c_x * \psi^m_{n\ell j}](\omega) = \widehat{\psi}^m_{n\ell}(2^j\omega) \sum_{k=1}^{N_x} c(Z_k) e^{-i\omega \cdot R_k}\,, \tag{2.41}$$

where $\mathcal{F}[h](\omega) = \widehat{h}(\omega)$ is the Fourier transform of the function $h \in \mathbf{L}^1(\mathbb{R}^3)$. The Fourier transform of $\psi^m_{n\ell}$ can be computed analytically:

$$\widehat{\psi}^m_{n\ell}(\omega) = (-i)^\ell \sqrt{\frac{4\pi}{2\ell+1}} |\omega|^{2(n-1)-\ell} e^{-\beta^2|\omega|^2/2} Y^m_\ell(\omega/|\omega|)$$

Therefore (2.41) can be evaluated directly for any $\omega \in \mathbb{R}^3$.

In particular we compute, via direct numerical evaluation, a tensor $\Psi^m_{n\ell j} \in \mathbb{C}^{L_x} \times \mathbb{C}^{L_x} \times \mathbb{C}^{L_x}$ defined as

$$\Psi^m_{n\ell j} = \mathcal{F}[\rho^c_x * \psi^m_{n\ell j}]\Big|_{\Omega_x} \tag{2.42}$$

Due to the discretization in (2.42), taking the inverse fast Fourier transform (iFFT) of $\Psi^m_{n\ell j}$ recovers the circular convolution $\rho^c_x \circledast \psi^m_{n\ell j}$ evaluated on the spatial grid $\mathcal{G}_x$:

$$\text{iFFT}(\Psi^m_{n\ell j}) = \rho_x \circledast \psi^m_{n\ell j}\Big|_{\mathcal{G}_x} \tag{2.43}$$

The direct computation of $\Psi^m_{n\ell j}$ requires $CN_xL_x^3$ floating point operations, whereas the iFFT calculation requires $CL_x^3 \log L_x$ floating point operations. Therefore the total cost is reduced to $O((N_x + \log L_x)L_x^3)$.

First order wavelet scattering features are estimated by applying the pointwise nonlinear operator σ to (2.43) and estimating the integrals with a Riemann sum approximation. Second order wavelet scattering features are computed by taking the fast Fourier transform (FFT) of $\sigma(\rho_x \circledast \psi_{n\ell j})\big|_{\mathcal{G}_x}$ and computing the second circular wavelet convolution via frequency multiplication with a direct evaluation of $\widehat{\psi}^{m_2}_{n_2,\ell_2}(2^{j_2}\omega)$ on the grid $\omega \in \Omega_x$, followed by another iFFT, application of σ, and Riemann sum. The cost of each second order feature, given that (2.43) must already be computed for the first order features, is $O(L_x^3 \log L_x)$.

Asymptotically one has $L_x^3 = O(N_x)$ which implies that the cost of calculating $\Psi^m_{n\ell j}$ is $O(N_x^2)$. We can reduce the cost further by replacing ρ^c_x with

$$\tilde{\rho}^c_x(u) = \sum_{k=1}^{N_x} c(Z_k)b(u - R_k) \tag{2.44}$$

where $b : \mathbb{R}^3 \to \mathbb{R}$ is a small, compactly supported bump function whose support only covers $\tilde{c}$ voxels. The cost of constructing $\tilde{\rho}_x(u)$ is $O(N_x)$. We then compute the FFT of $\tilde{\rho}_x(u)$, yielding $\hat{\tilde{\rho}}_x(\omega)$, in $O(L_x^3 \text{log} L_x) = O(N_x \text{log} N_x)$ time. We then obtain Equation (2.41), and hence $\Psi^m_{n\ell j}$, by computing $\hat{\tilde{\rho}}_x(\omega)\hat{\psi}^m_{n\ell j}(\omega)$ for all $\omega \in \Omega_x$, which costs $O(L_x^3) = O(N_x)$ computations. Thus the total computational cost is

$$O(N_x) + O(N_x \text{log} N_x) + O(N_x) = O(N_x \text{log} N_x) \tag{2.45}$$

which is smaller than $O(N_x^2)$.

One final consideration to note when examining the scaling is that the number of features is not constant with the number of atoms. The number of first layer wavelets is the number of (n, ℓ) pairs

times the number of scales J. Of these, only J scales with N_x, and it scales as

$$J = O(\log L_x^3) = O(\log N_x) \tag{2.46}$$

Thus the number of floating point operations for the first layer is

$$O(N_x \log N_x) \times O(\log N_x) = O(N_x (\log N_x)^2) \tag{2.47}$$

In the second layer there are $O(J^2) = O((\log N_x)^2)$ wavelets, so the additional computational complexity for the second layer is

$$O(N_x (\log N_x)^3) \tag{2.48}$$

However, for large atomic structures we do not need J to be large enough for the biggest wavelet to cover the entire structure, but only so that the biggest wavelet covers the largest distance of relevant interaction between two atoms. So for systems beyond a certain size, we may be able to avoid the additional factors of $\log(N_x)$ for the scaling.

The reason that we do not elect to compute this more efficient algorithm is that there is an error induced by taking the FFT of $\tilde{\rho}_x(u)$. Small bumps in space will have a large frequency support. Taking the FFT will wrap the high frequency portions around the cube creating interference. It is possible that this issue could be mitigated to some degree by controlling the smoothness of the bumps, but it cannot be eliminated since anything that is bounded in space has infinite frequency support. By defining our pseudo density in frequency, we avoid this problem albeit at the cost of missing an opportunity for improved scaling.

2.6 Prior Work on Scattering Transforms

The original scattering transform for general d-dimensional signals was introduced by Mallat in [49]. In that version of the scattering transform, the wavelets were formulated in Cartesian coordinates and additional mother wavelets were generated via the d-dimensional rotation group action. Furthermore, the original scattering transform used the standard complex modulus as its nonlinearity, not the special modulus, and was designed to be translation invariant. In [49] it was also shown that the scattering transform is stable to diffeomorphism group actions.

Subsequent theoretical work on the scattering transform was developed in [50, 51, 52, 53]. Scattering transforms have been applied to problems in audio signal processing [54, 55, 56, 57, 58] and image processing [59, 60, 61, 62, 63, 64, 65, 66]. The development of three-dimensional scattering transforms for quantum chemistry was initiated in [67, 68]. Subsequent developments, with a focus on applications to small organic molecules in equilibrium position, are contained in [43, 44].

CHAPTER 3

LEARNING A MODEL

3.1 Linear Regression

With the wavelet scattering coefficients computed as a set of features that forms a rich representation of an atomic system, what remains is to combine the information between them to form a prediction of the energy. We elect to do this by using a linear regression. We would like to choose the simplest model to combine the features in the expectation that this will improve the generalizability. We first propose a model of the form

$$\tilde{E}(x;w) = \sum_{c,q} w_{c,q} \|\rho_x^c\|_q^q + \sum_{c,\lambda,q} w_{c,\lambda,q} \frac{||\sigma(\rho_x^c * \psi_\lambda)||_q^q}{||\rho_x^c||_q^q} + \sum_{c,\lambda_1,\lambda_2,q} w_{c,\lambda_1,\lambda_2,q} \frac{||\sigma(\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2})||_q^q}{||\sigma(\rho_x^c * \psi_{\lambda_1})||_q^q}$$

Here we divide each feature by the corresponding feature of one lower order to remove the dependence on the number of atoms. This is only necessary when predicting a per atom quantity. We simplify the denominator later. Taking a mean squared loss function, we would like to solve for weights such that

$$w = \arg\inf \left[\frac{1}{n_t} \sum_{i=1}^{n_t} |E(x_i) - \tilde{E}(x_i;w)|^2 \right] \tag{3.1}$$

for n_t data points in the training set. For now let us define the training feature matrix as $\Phi \in n_t \times |\Gamma|$ for notational simplicity. The selection of the weights can be solved for via closed form

$$w = (\Phi^T \Phi)^{-1} \Phi^T E(X), \tag{3.2}$$

where $X = \{x_i\}_{i=1}^{n_t}$.

However, there are several reasons why a simple least squares regression does not work well in this case. Since some sets of features are computed similarly (ie same wavelet with different q), the matrix of all features can have highly correlated columns. This means that the conditioning is

poor, and an inversion is not possible. Even when computing the weights via a reasonably chosen decomposition, the set of non-regularized weights result in poor off-training predictions. We thus need some type of regularization to reduce model volatility. We would also like the model to make fast predictions once the weights are learned. If we can only compute a small set of the available features for future data points, we can speed up computation significantly. For these reasons, we use an ℓ_0 regularized (sparse) regression, and selecting the maximum number of nonzero weights as M, we rewrite Equation (3.1) as

$$w = \arg\inf_{\tilde{w}} \left[\frac{1}{n_t} \sum_{i=1}^{n_t} |E(x_i) - \tilde{E}(x_i; \tilde{w})|^2 : ||\tilde{w}||_0 \leq M \right] \tag{3.3}$$

Solving for the weights in an ℓ_0 regression is NP-hard. Instead we use the iterative (greedy) orthogonal least squares method selecting one feature for inclusion at a time [69]. We begin by normalizing the features to have unit variance across the training data set. Then we select the feature with maximal inner product against the normalized energy.

$$\phi_k = \arg\sup_{\gamma \in \Gamma} \left\langle \phi_\gamma, E(X) \right\rangle \tag{3.4}$$

We decorrelate the remaining features with respect to the selected feature ϕ_k

$$\tilde{\phi_\gamma} = \phi_\gamma - \left\langle \phi_\gamma, \phi_k \right\rangle \phi_k \tag{3.5}$$

We subtract the selected feature's contribution from the energy.

$$E_{next}(X) = E_{prev}(X) - \left\langle E_{prev}(X), \phi_k \right\rangle \phi_k \tag{3.6}$$

After this, we re-normalize the remaining features and energy to have unit variance, and repeat the selection process up to M_{max} features. With the selected set of features, we set a new feature matrix which contains only the M_{max} selected columns of our original feature matrix. We then compute the QR factorization of this matrix.

$$\tilde{\Phi} = \begin{bmatrix} \phi_{k_1}(x_1) & \phi_{k_2}(x_1) & \dots & \phi_{k_{M_{max}}}(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{k_1}(x_{n_t}) & \phi_{k_1}(x_{n_t}) & \dots & \phi_{k_{M_{max}}}(x_{n_t}) \end{bmatrix} = QR$$

where n_t is the number of training points and $[\phi_{k_1}, \ldots, \phi_{k_{M_{max}}}]$ are the greedily selected features.

We perform 5-fold cross validation to ensure that our results are representative of the whole data set. In each fold a different 20% of the data is held out for testing. With the remaining 80% in each fold, we further divide the data and remove 20% as validation data. The weights are learned on the training data, the remaining 60% of the data in each split.

We will have a unique model for each number of features included in the regression. Since M_{max} can be chosen arbitrarily, we would like to find an M which minimizes the error on the test set without using the test set. The error on the training set is minimized at $M = M_{max}$, but for large M_{max}, the model may be overfitted to the training data resulting in a worse test error than a simpler model. So instead, we use a set of validation data which are held out from the training up to this point. On average, the M which minimizes the error on the validation set will also minimize the error on the test set assuming the validation set is sampled from the same distribution as the test set. For each possible M, we take the first M columns of the Q matrix and solve for weights as in Equation (3.1). Since Q has orthonormal columns we have

$$w_Q = Q^T E(X) \tag{3.7}$$

We then compute a pseudo Q matrix for the validation data points. Note that this will only return reasonable results if the validation data is drawn from the same distribution as the training data.

$$Q_{valid} = R^{-1}\Phi_{valid} \tag{3.8}$$

We restrict the columns of Q_{valid} to the first M and apply the weights we learned from the training data. Our model is then selected as

$$M^* = \arg\inf_M \left[\frac{1}{n_{val}} \sum_{i=1}^{n_{val}} |E(x_i) - \tilde{E}(x_i; w_Q)|^2 : \|w_Q\|_0 \leq M \right] \tag{3.9}$$

To make energy predictions from the test data, we have to compute a Q_{test} the same as we did for Q_{valid}, again counting on the test data being drawn from the same distribution as the training data. We then apply the model learned from the training and validation data and get a test data

energy prediction. For each test fold, we have four different training/validation splits of the non-test data. Each of these splits gives a unique model. Averaging over these four models improves the prediction on the test data. Averaging the errors on each of the five test folds gives our final error results.

3.2 Data

The training, validation, and interpolation testing data for the machine-learned model consists of amorphous $Li_\alpha Si$ structures labeled by formation energies calculated using Density Functional Theory (DFT). These structures are in cubic boxes under periodic boundary conditions containing from 55 to 100 atoms, with lithium-to-silicon ratio α ranging from 0.1 to 3.75. Initial disordered structures are generated by evolving random structures under ReaxFF [70] molecular dynamics (MD) at 2500K for 10 ps, and ten different disordered structures are randomly picked from the MD trajectory for each of the 37 chosen concentrations. The accuracy of the force field used to obtain the initial amorphous structure is not important, due to the following DFT calculations. In particular, each structure is fully relaxed at constant volume using DFT allowing the energy to decrease to a stable minimum. The structures and formation energies along the relaxation paths make up the amorphous dataset used in this work, which contains a total of 90,252 structures. A histogram of the quantity of these structures by energy and concentration is shown in Figure 3.1. We note that the structures are heavily concentrated near the endpoint of the relaxation, so we expect the resulting model to do better on near-equilibrium amorphous structures. This is desirable because the low-energy structures are more likely to arise in realistic simulations. We also calculate voltages versus Li/Li^+ [71, 72, 73] and radial distribution functions for Si-Si, Li-Si, and Li-Li pairs, and find good agreement with similar data sets from the literature [74, 75] This confirms that our amorphous structures are physically realistic.

Formation energies and relaxations were performed in the Vienna Ab initio Simulation Package (VASP) using the Projector-Augmented Wave method and the PBE exchange-correlation functional with a plane-wave energy cutoff of 500 eV. The Brillouin zone was sampled using the Gamma point

only during relaxation. After relaxation, the energies along each relaxation path were corrected for k-point sampling errors by calculating the energy of each fully relaxed structure using a $3 \times 3 \times 3$ Gamma-centered grid and applying the resulting constant shift to the rest of the structures in the relaxation path. The mean absolute k-point sampling correction was 27 meV/atom. The total

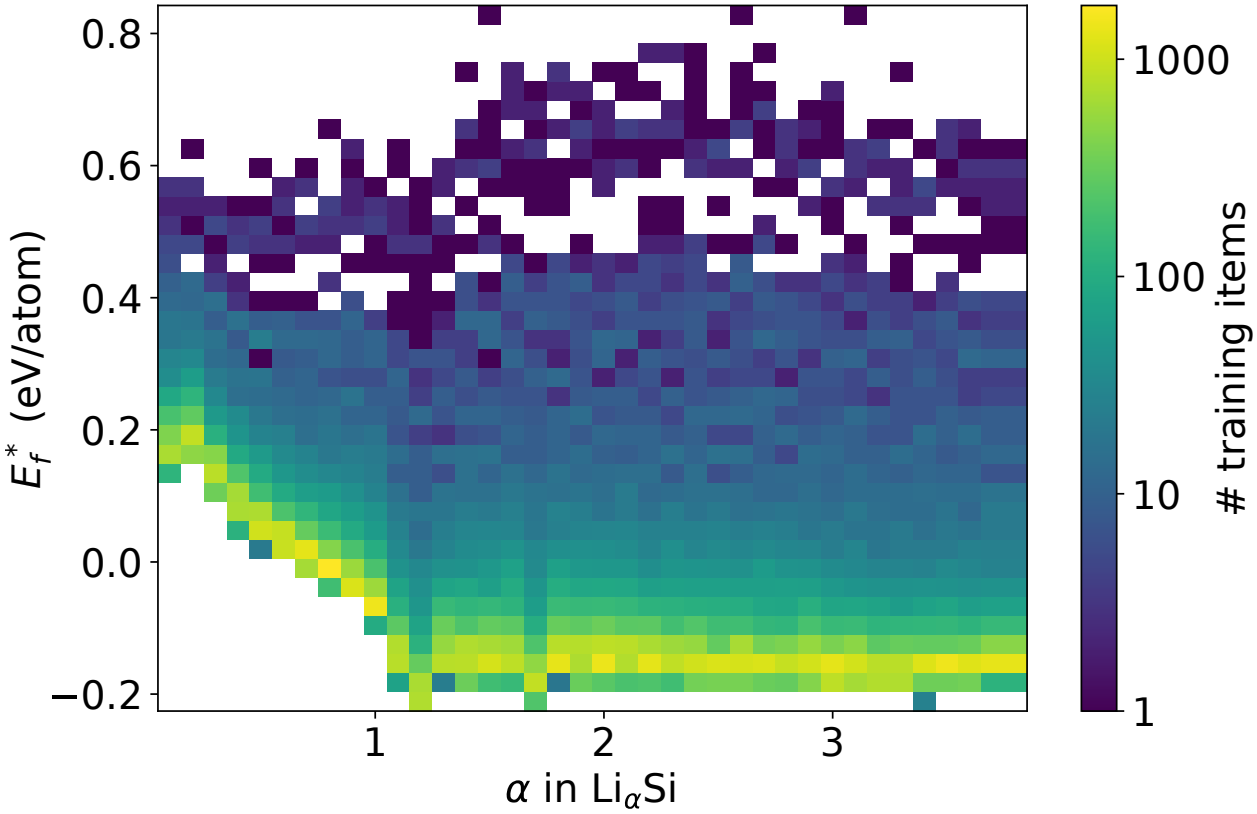

Figure 3.1: Histogram of training set energies versus concentration α in $Li_\alpha Si$. Color indicates the number of training items in each bin on a logarithmic scale.

formation energy of a structure with $\mathrm{N_{Li}}$ lithium atoms and $\mathrm{N_{Si}}$ silicon atoms is defined based on DFT total energies:

$$E_f(\mathrm{Li_{N_{Li}}Si_{N_{Si}}}) = E_1(\mathrm{Li_{N_{Li}}Si_{N_{Si}}}) - \mathrm{N_{Li}}E(\mathrm{Li}) - \mathrm{N_{Si}}E(\mathrm{Si})\,,$$

where $E_1(\mathrm{Li_{N_{Li}}Si_{N_{Si}}})$ is the total energy of the system, and $E(\mathrm{Li})$ and $E(\mathrm{Si})$ are the DFT total energy per atom of elemental lithium and silicon, respectively. The structure $\mathrm{Li_{N_{Li}}Si_{N_{Si}}}$ has reduced formula $\mathrm{Li}_\alpha\mathrm{Si}$ with $\alpha = \mathrm{N_{Li}/N_{Si}}$ and per-atom formation energy

$$E_f^*(\mathrm{Li}_\alpha\mathrm{Si}) = E_f(\mathrm{Li_{N_{Li}}Si_{N_{Si}}})/(\mathrm{N_{Li}} + \mathrm{N_{Si}})\,. \tag{3.10}$$

The per-atom formation energy is the quantity of interest for machine learning. Notice, though, it includes the terms $N_{Li}E(Li)$ and $N_{Si}E(Si)$ which require no additional quantum mechanical calculations beyond the one-time cost of computing $E(Li)$ and $E(Si)$. The difficulty is in computing $E_{tot}(Li_{N_{Li}}Si_{N_{Si}})$, which requires a costly DFT calculation for each new state. When fitting our machine learned models, we regress the per-atom total energy, defined as:

$$E^*_{tot}(Li_\alpha Si) = E_{tot}(Li_{N_{Li}}Si_{N_{Si}})/(N_{Li}+N_{Si})$$

or

$$E^*_{tot}(Li_\alpha Si) = E^*_f(Li_\alpha Si) + \frac{\alpha}{1+\alpha}E(Li) + \frac{1}{1+\alpha}E(Si)\,.$$

Even though it is simple to convert total energies into per-atom total energies, we regress the latter since per atom energies remove the effect of varying unit cell sizes and the number of atoms per unit cell on the total energy. Since we use the squared loss as our measure of error when training, regressing total energies would bias the models towards systems containing larger numbers of atoms since the total energy scales with the number of atoms.

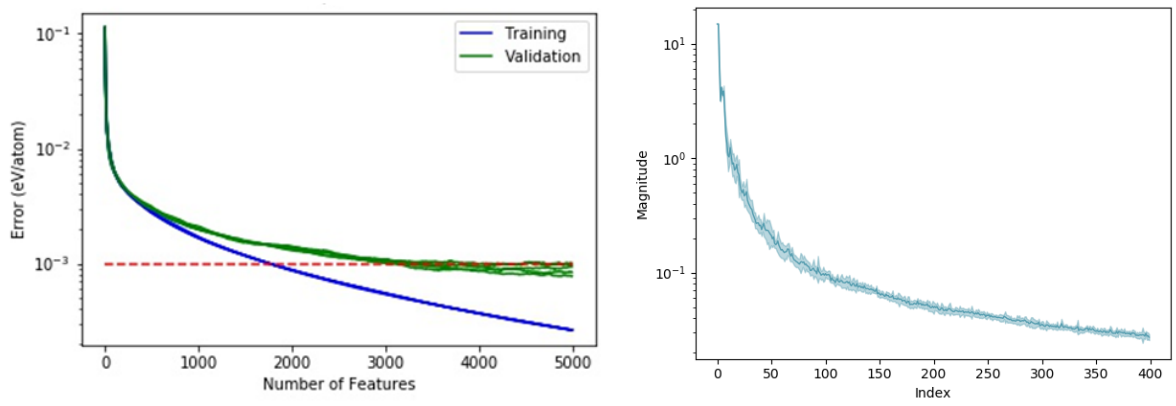

Figure 3.2: Left: Regression RMSE vs model size. Left: weights vs model size

There are two publications which train neural networks on similar lithium silicon data sets [74, 76], but we were unable to attain their data to be able to compare results precisely. Comparisons to these works seem reasonable, but the difference in that exact data being used could lead to significantly different results.

3.3 Results

The precision of the formation energy of the data structures is 1.0 meV/atom. That level designates our target error as anything below it constitutes learning the particular version of DFT used to generate the data rather than the actual chemistry. There are several hyperparameters that we select to set the collection of features. We use five density channels: lithium, silicon, valence, ionic, and kinetic which we define for lithium (3) and silicon (14).

$$c_l(3) = 3, \quad c_l(14) = 0$$

$$c_s(3) = 0, \quad c_s(14) = 14$$

$$c_v(3) = 1 \quad c_v(14) = 4$$

$$c_i(3) = 2 \quad c_i(14) = 10$$

$$c_k(3) = \sqrt{3} \quad c_k(14) = \sqrt{14}$$

The lithium and silicon channels partition the state x along atom species, whereas the valence and core channels separate the state x according to electron type. The kinetic channel gives us a scaling of the charge that matches the kinetic portion of the energy. We use six wavelet scales $j = [0, \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}]$, the wavelets with $1 \leq n \leq 5$, associated ℓ values, and $q = \{1, 4/3, 5/3, 2\}$. The second order scattering coefficients were restricted so that $j_2 \geq j_1$ and $(n_2, \ell_2) > (n_1, \ell_1)$.

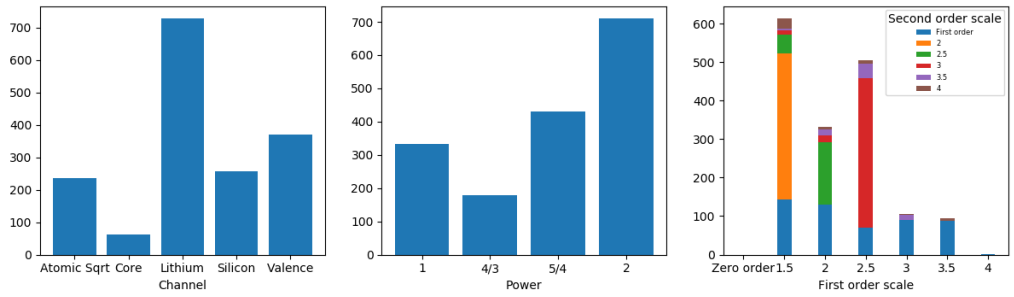

Figure 3.3: Statistics of selected wavelet coefficients. From list to right: channel c, power q, scale pairing (j_1, j_2).

Using first-order features exclusively, we obtained a mean absolute error of 2.1 meV/atom [77]. This error is significantly lower than current neural network models trained on similar data sets

[74, 76]. We see a significant improvement in performance when we include second-order features in the model, reaching an MAE of 0.78 meV/atom. As can be seen in Figure 3.2 the error on the training set decreases approximately as $1/M$ for M features included in the model. The validation set error closely follows the training error which demonstrates that we are not learning only spurious noise from the data, but that our model encodes chemical information. Even with a very sparse model - $M = 100$ - we still achieve a reasonable RMSE of 5.8 meV/atom. We also see from the figure that the weights decay very quickly.

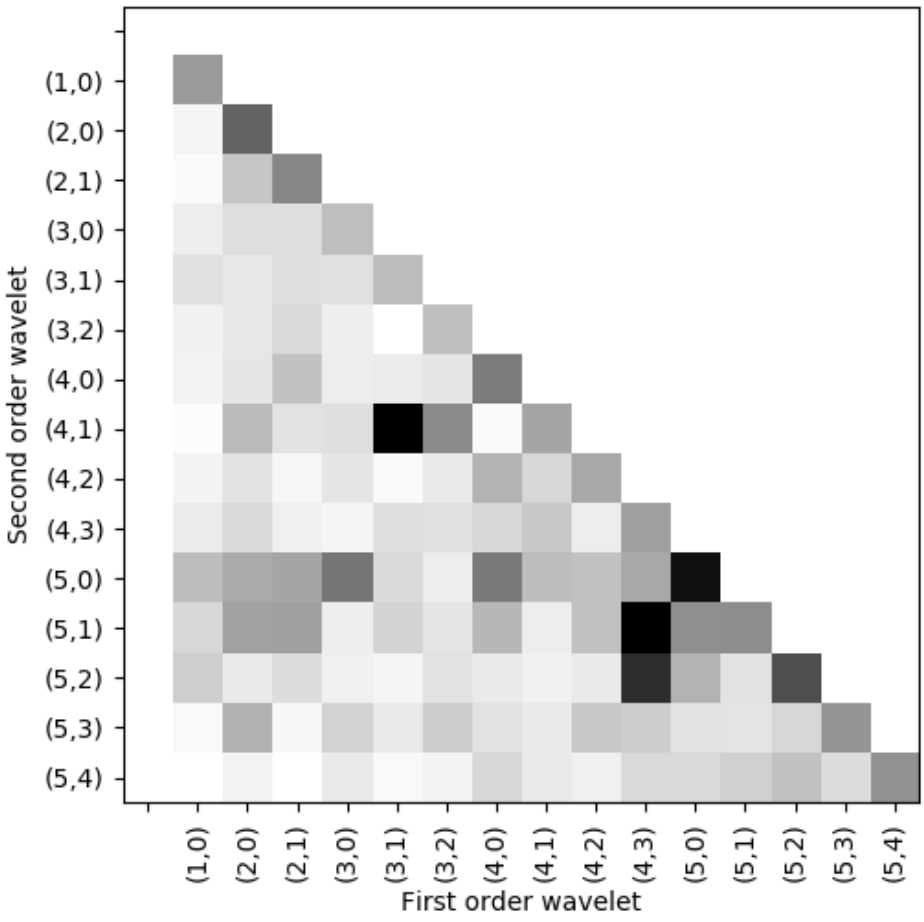

Figure 3.4: Histogram of training set energies versus concentration α in $Li_\alpha Si$. Color indicates the number of training items in each bin on a logarithmic scale.

Since our model is sparse, we can analyze the selected features to see what aspects made a

representation useful for the energy prediction. In Figure 3.3, we see that among the channels, the lithium channel was selected most commonly. Representations in the lithium channel contain no information of the silicon atoms, so this tells us that there is substantial information in the distribution of the lithium atoms. This seems reasonable since the lithium concentration varies throughout the data set and characterizes aspects of the structures properties. The four integration powers are all included in the model suggesting that they are complementary in the information they contain. In the scale plot, we see that large scales are not included very often, and in particular, our model would be no worse off if we removed the largest scale. We see far more second order features with coupled scales than large scale features. The second order features encode a more refined wide reaching feature as the wavelet that aggregates local information does not have to match the wavelet which aggregates the neighborhoods. Figure 3.4 gives us some additional information, breaking the features into bins according to their corresponding wavelet (n, ℓ) pairing. We can use this to reduce our computation so that only the most useful features are computed.

CHAPTER 4

IMPROVING GENERALIZABILITY

4.1 Alteration of Our Previous Model

Recall from Chapter 3 and Figure 3.1 we have a training database of 90,252 amorphous $Li_\alpha Si$ structures with DFT computed energies spread across 37 concentrations α, ranging from 0.1 to 3.75. These structures correspond to 370 relaxation paths beginning at an initial set of 370 high energy states, with 10 relaxation paths per concentration. In this section, we make some modifications to our previous model. To simplify the form of the features used in the regression, we divide them by the number of atoms of each state instead of by the norm of the lower order features.

$$\begin{aligned}\tilde{E}(x;w) = &\sum_{c,q} w_{c,q} \frac{\|\rho_x^c\|_q^q}{N_x} + \sum_{c,\lambda,q} w_{c,\lambda,q} \frac{||\sigma(\rho_x^c * \psi_\lambda)||_q^q}{N_x} \\ &+ \sum_{c,\lambda_1,\lambda_2,q} w_{c,\lambda_1,\lambda_2,q} \frac{||\sigma(\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2})||_q^q}{N_x}\end{aligned} \tag{4.1}$$

While we achieve great predictive ability with the model from Chapter 3, there are several directions in which we can improve it. Sampling data uniformly across the data is standard practice, including in the machine learning for quantum chemistry field. However, the lithium-silicon data set is time series data, and even though we are not making predictions over time, care must be taken in the data sampling. There are many atomic systems which are extremely close, particularly in the low energy portion of the training data. This gives an advantage in "generalizability" because if a particular structure is set aside for testing, it is likely that we will include at least one other that is only a slight deviation away in the training data.

Additionally, this sampling is unnatural to the data generation. The goal of machine learning in quantum chemistry is to reduce the computational cost. If we require that the energy of every structure in every relaxation path is computed, nothing is saved. Even though all of the lithium

silicon data has already been processed, we would like to demonstrate a method where we can specify some portion that did not have to be computed for our method to work.

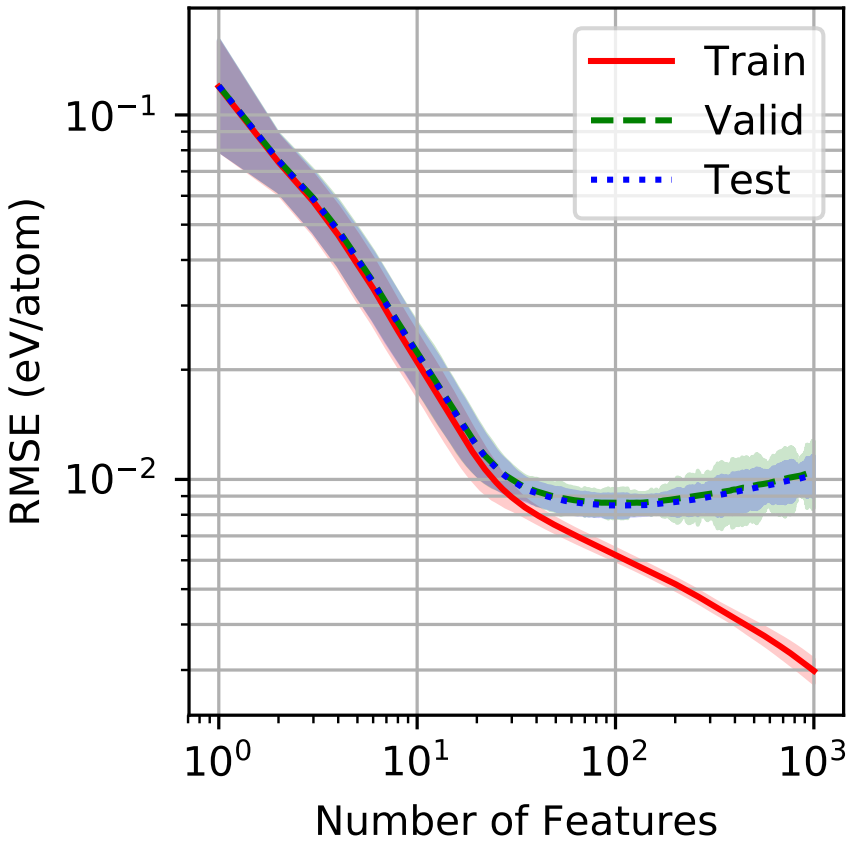

Figure 4.1: Errors on the amorphous $Li_\alpha Si$ database as a function of number of features included in the model on a log-log scale. Error on the training set is shown in red, the validation set is shown in green, and the test set is shown in blue. The training error is a decreasing function of the number of features, whereas the validation and testing curves are not. The value $M^\star$ that minimizes the validation curve is the algorithm's best estimate for the optimal model that best balances under- and over-fitting of the training data. It has good agreement with the minimum of the test error curve.

There are two options which seem amenable to this goal. Since the simulations of the structures begin at high energy and decrease to an equilibrium, one option is to take the high energy portion only. The issue with this approach is that the model would be expected to make many energy predictions below the energy range on which it is trained. It is unreasonable to expect good performance. The other option is to sample across the relaxation paths. There are two advantages to this form of sampling. It only requires the costly chemical computations to be done on a select

set of paths, and it displays a truer generalizability.

Using five-fold cross validation, we randomly partition the relaxation paths into five sets of 74 relaxation paths with two paths per concentration in each of the sets of 74. We place four of these sets, 296 relaxation paths total, in the training/validation set, and one set of 74 paths in the test set. We rotate through using each set separately as a test set, meaning that we carry out all numerical experiments five times, each time with a different training/validation and test set split. Empirical results indicate this training paradigm significantly restricts the degree to which the machine learned model can fit non-physical spurious patterns in the data. We leave for future work developing a model that can predict the entire relaxation path starting with the only the highest energy state.

We augment the learning process by leveraging empirical bootstrapping and feature bagging [78]. Given an initial database of n_t states and their energies (that does not include the withheld testing set), the empirical bootstrap algorithm samples the database with replacement n_t times to obtain the training set. Those states not selected for the training set are placed in the validation set. This approach allows us to construct many different models from one database, which are then averaged. The resulting averaged model, which is still a linear model over the representation $\Phi(x)$, is superior to any one individually fitted model since the averaging reduces random fluctuations in the fitting process that result from spurious patterns in a single training set. In order for this averaging process to have maximum effect, the weights of the individual models must be as uncorrelated as possible.

Feature bagging, which is a prominent component of random forests, decorrelates the models by restricting the greedy selection at each greedy step. In particular, at each greedy step in the OLS algorithm, approximately $\sqrt{d+1}$ features are sampled without replacement from among the full set of d features in $\Phi(x)$ plus the bias term, minus the features that have already been selected up to that point. The OLS algorithm at each step must then select from among the sampled features, which due to the randomness in the feature sampling, results in models that are significantly less correlated. Indeed, in our own numerical experiments, the most significant features selected with empirical bootstrapping, but without feature bagging, are very often identical. While restricting

the number of possible features at each greedy step means that each model has larger error on the training set, the aggregated average model improves on the test set [78].

With the 296 training relaxation paths, we carry out the model fitting algorithm described in Section 3.1 jointly with the bootstrapping and bagging. For the training set, we randomly select according to a uniform distribution, with replacement, 296 relaxation paths from the training set. Those paths selected more than once are repeated in the training set with the number of copies equalling the number of times the path was selected. Those paths that are not selected are placed in the validation set. The sparse linear model is trained using the greedy OLS algorithm with randomized feature bagging, with the number of features M ranging from $M = 1$ to $M = M_{\max} = 1000$. We use $n = 3$, $J = 4$, $q \in \{1, \frac{4}{3}, \frac{5}{3}, 2\}$, and the five channels previously discussed. The optimal number of features $M = M^\star$ is selected by minimizing the loss on the validation set. This procedure is repeated 100 times, resulting in 100 sparse linear models of the form (4.1), which are averaged together to yield the final model.

	RMSE (meV/atom)	MAE (meV/atom)
Relaxation paths	7.44 ± 0.49	5.52 ± 0.34
Diffusion	12.3 ± 0.50	11.7 ± 0.51
Large states	9.54 ± 0.25	6.81 ± 0.23
Bulk modulus	12.8 ± 1.36	8.92 ± 0.68

Table 4.1: Numerical results for ML predictions on the test data from the amorphous dataset and the three extrapolation tasks from the model trained only on the amorphous data.

This final model is evaluated on the withheld test set. Figure 4.1 depicts the training, validation, and testing errors as a function of the number of model features M. It indicates that best models have, generally, between 64–256 features, with an average of 121 features per model, a small number given that there are approximately 70,000 training structures. Furthermore, the validation curve closely follows the test curve, indicating that our cross-validation procedure is nearly optimal for this test data. The average root mean squared error (RMSE) and the average mean absolute error (MAE) over the five test folds, along with the standard deviation, is reported in the first row (relaxation paths) of Table 4.1. Despite the small number of features, the RMSE is 7.44

meV/atom and the MAE is 5.52 meV/atom, which is comparable to the results reported in [76] and [74], both of which trained neural networks on lithium silicon data, and is small enough to be of use in materials science applications. However, the model developed here is significantly simpler than neural network models, being a linear model over multiscale, invariant features that utilize a universal set of filters. As such, the model is adept at generalization, as reported in the next section.

We notice some significant differences between the results in Figure 4.1 and those in Chapter 3 on the left side of Figure 3.2. In the previous model, the validation curve never has a positive slope, whereas for the model in this chapter, we see a divergence of the validation and training curves at only 100 features. This has significant ramifications going forward. The models have similar results for the training error indicating that our representation has good flexibility to be able to fit the data. The difference in the validation error sheds light on the distribution of the lithium-silicon data. When we randomly sample across all points, the validation and testing data are extremely similar to the training data. When we sample between relaxation paths, the validation and testing differ from the training data, and importantly, they differ in approximately the same way leading to both reaching a minimum error at around 100 features. This is important for the following section because we need the validation data to select a number of features that will lead to a model that generalizes well.

4.2 Extrapolation Data

In order to test the machine learning model's generalizability to extrapolation tasks, additional DFT data are required to compare with the results of the machine learning model. We test three different extrapolation tasks: prediction of migration barriers, energy prediction for systems with larger unit cells, and prediction of elastic properties.

4.2.1 Diffusion Barriers

Diffusion barriers cannot be defined uniquely in amorphous structures due to the lack of order. Rather, paths that move an atom from one favorable coordination environment to another through

a relatively unfavorable environment are abundant. An endpoint for such a pathway was found by locating void spaces in the amorphous structure through Voronoi analysis and inserting a test lithium atom at each void to find the most energetically favorable position. Nearby lithium atoms to this void were subsequently identified, and the minimum-energy path (MEP) for each lithium to travel to the void was calculated using the nudged elastic band (NEB) method [79]. The NEB images along 6 calculated MEPs (for a total of 50 image structures) were used as testing data for this extrapolation task. The primary quantity of interest is the migration barrier, i.e., the difference between the lowest-energy and highest-energy points along the MEP.

One important application of atomistic simulation is the study of atomic migration from site to site. The energetic barrier to migration determines diffusion constants and ionic conductivity. The diffusion process may be simulated by directly tracking the mean square displacement using molecular dynamics or by calculating the migration barrier and using the Nernst–Einstein relationship. The first step in the explicit calculation is to find the minimum energy path (MEP) for an atom to travel between two stable sites. This is typically done using optimization techniques such as the Nudged Elastic Band (NEB) method. The barrier is defined as the energy difference between the stable position and the highest-energy position (saddle point) along the MEP.

There are a number of reasons why calculation of diffusion barriers may present a challenge for our ML model. Our present models do not predict forces, so they cannot be used with the NEB for prediction of the path itself. We therefore simply predict energies along the DFT-calculated MEP. A more fundamental challenge is the fact that the transition state structure, with one atom in a high-energy state and the rest in relatively low-energy states, is qualitatively different from the training items in the amorphous $Li_{\alpha}Si$ dataset. Calculation of diffusion barrier is thus an extrapolation task. Furthermore, there is only one diffusing atom in the simulation box during calculation of the diffusion barrier. This means that energy per atom is no longer the most relevant measure of error. Instead, total energy differences between similar structures along the MEP are the relevant quantity. Cancellation of systematic errors in DFT allows the calculation of energy differences along diffusion paths with much higher accuracy than would be suggested based on the

accuracy of the method in total energy per atom [80]. It remains to be seen if similar cancellation of errors can improve the accuracy of diffusion barriers predicted by an ML model.

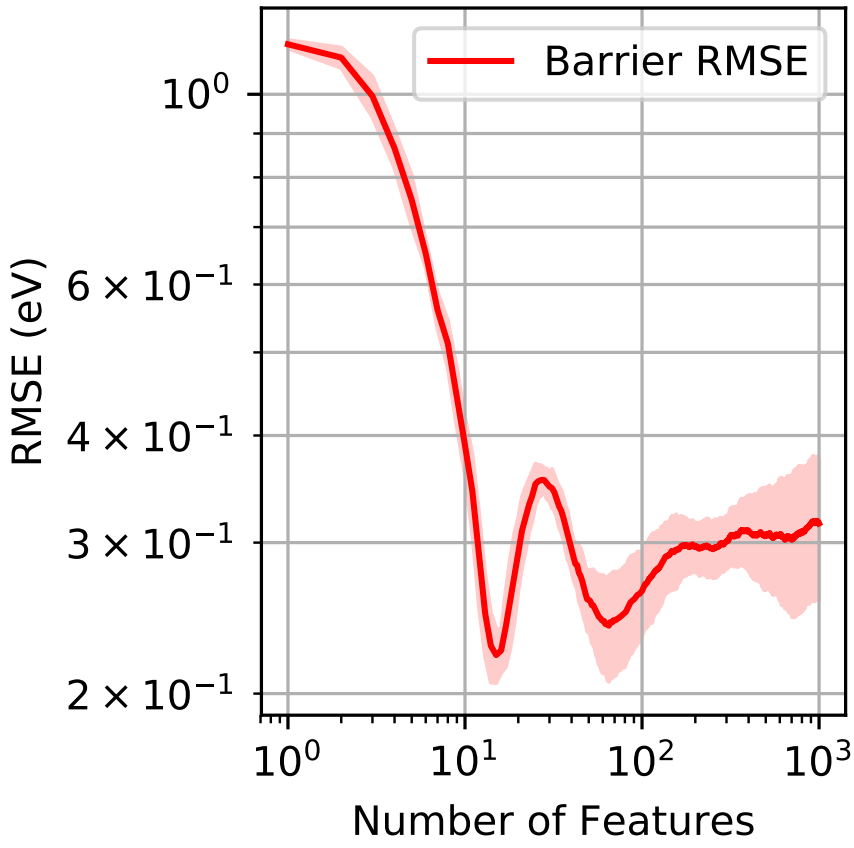

Figure 4.2: Log-log plot of RMSE in diffusion barrier prediction averaged over the five folds.

To test the extrapolation of our model to diffusion barriers, void spaces were identified in $Li_{0.2}Si$ and $Li_{0.5}Si$ by Voronoi analysis. Candidate endpoint structures were created by moving nearby lithium atoms into the voids and relaxing the resulting structure while keeping the target lithium atom fixed. Six endpoints were identified in which the void space was a local optimum for the lithium atom and in which the relaxation for the rest of the structure was minimal. These endpoints were then used together with the original $Li_\alpha Si$ structures as the basis for NEB calculations. The structures along the resulting NEB path were then passed to the ML model for comparison with the DFT results.

The learning curves are shown in Figure 4.2. The RMSE for the diffusion path structures is

less smooth than the RMSE for test folds consisting of the relaxation paths in the amorphous $Li_\alpha Si$ data. Table 4.1, second row (diffusion), shows the RMSE and MAE of the per atom energy across all diffusion barrier structures. The RMSE for these structures is about 12.3 meV/atom, which is worse than on the relaxation path test but by less than a factor of two. Nevertheless, reduced accuracy is expected given the extrapolative nature of the task.

Path	Barrier (ML Model)	Barrier (DFT)
1	0.228	0.226
2	0.819	0.341
3	2.256	2.139
4	0.230	0.402
5	2.613	2.224
6	0.326	0.354

Table 4.2: Diffusion barriers (in eV) along various paths as predicted by our ML model and DFT. Paths 1-5 start from the same $Li_{0.2}Si$ structure and path 6 is in $Li_{0.5}Si$.

However, these errors are not the diffusion barrier errors, which is the quantity of interest. The energies along the diffusion paths are shown in Figure 4.3. The first row plots the absolute energies for both the DFT calculation and the model prediction. The second row shifts the DFT and predicted energy curves to both start at zero, to more easily compare and read off the barriers, which are given in Table 4.2. The third row of Figure 4.3 plots the predicted energy curves as a function of the number of model features M, showing the learning rate of the model with respect to this task. The plots indicate that even with a small number of features, for example, $M = 21$ or $M = 41$, the energy curve and resulting barrier is qualitatively correct, with additional features serving to refine the curves and better align the total energies.

Visual inspection of the energy along the diffusion paths shows that much of the error is systematic. The $Li_{0.2}Si$ structures contain 60 atoms, so 12.3 meV/atom corresponds to 0.74 eV in total energy. If these errors were random, we would expect at least 0.74 eV error in prediction of the diffusion barrier. However, the curves show that the ML model can successfully distinguish between small-barrier paths and large-barrier paths, and the MAE in barrier prediction is 0.20 eV. While there is certainly room for improvement, we believe that these data show evidence that the

ML model is able to partially capture the physics involved in the diffusion process.

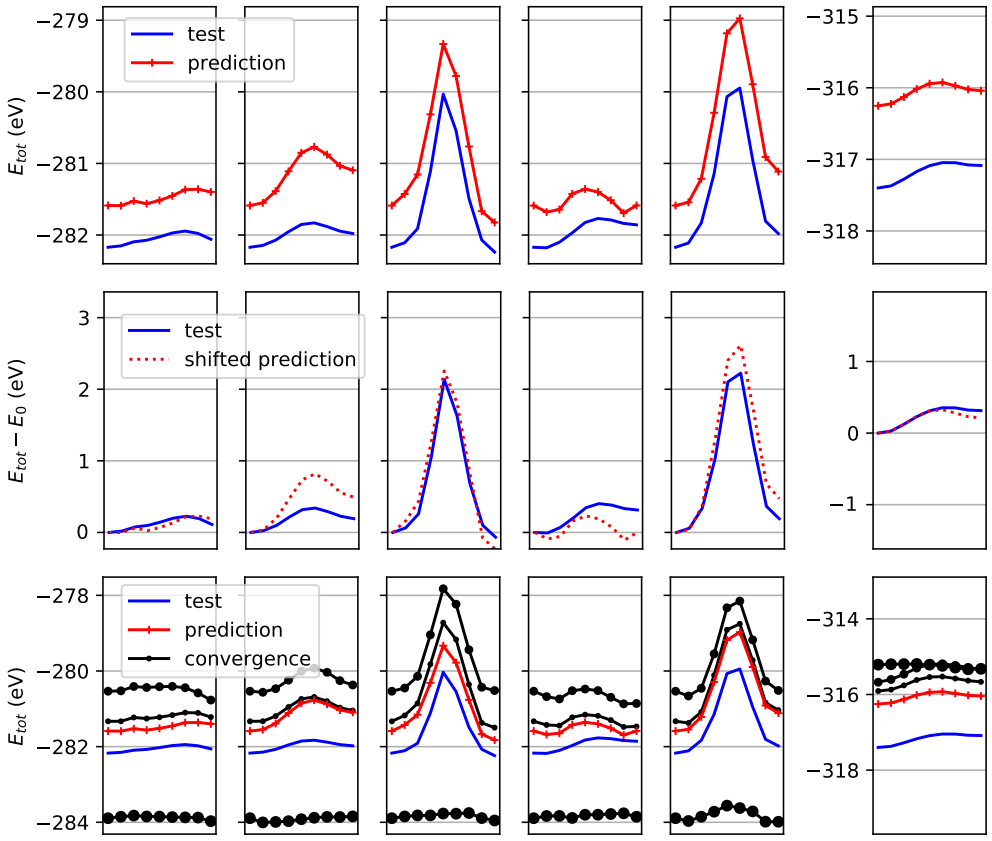

Figure 4.3: Plots of the six diffusion barrier paths (blue) and (top row) model predictions in red, (middle row) model predictions and test data shifted by their respective starting-point energies E_0, and (bottom row) convergence of models with increasing number of features used for predictions of diffusion barrier curves. The large radii circles coincide with fewer features used starting from a model with a single feature. The models quickly converge in shape and progress towards the red curve which is the aggregate model prediction. There is a curve for each choice of number of features $M \in \{1, 21, 41\}$.

4.2.2 Large Structures

For a second extrapolation task, we consider testing on structures that are significantly larger than the structures in the training set. Large structure testing data were generated by two methods: independently relaxing larger AIMD-generated structures (the "from-scratch" method) or tiling structures from the dataset, randomly perturbing all atomic positions by 0.1 Å, and performing

a single-point calculation (the "tiled" approach). The testing data consist of 37 from-scratch structures, 40 2 × 2 × 2 tiled structures, and 108 2 × 1 × 1 tiled structures.

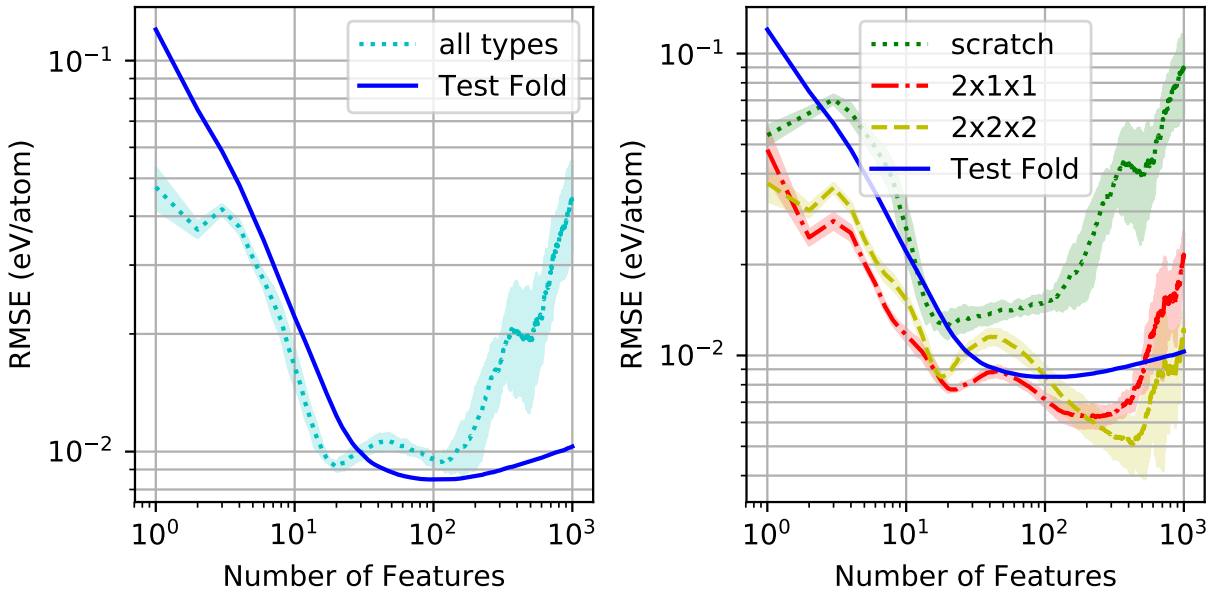

Figure 4.4: A log - log plot of average of RMSEs of models on the interpolation test set and on all types of large states (scratch, 2x1x1, 2x2x2). Here, y-axis = log(eV/atom), x-axis = log(number of features in models). The curves labeled 2x1x1, 2x2x2, and scratch on right are the RMSE of energy error predictions of the 5 aggregate models separated by test folds. On the left panel, we see that the location of the minimum (i.e., the optimal number of features) for the interpolation test error is similar to the optimal number of features for the extrapolation error on larger states, although model over-fitting is significantly more costly for the larger states' predictions.

It is desirable for an energy-predictor to generalize to structures in simulation cells with a different size than the training set so that it can be applied to simulation cells large enough to contain the geometries of experimental interest. As system size increases, the computation becomes challenging to carry out with DFT, but the wavelet scattering transform and linear regression scales efficiently with system size, and we are thus much less inhibited by large systems.

As discussed previously, the data for this task were generated by two different methods: "from scratch" and "tiled." The learning curves for each are shown in the right panel of Figure 4.2.2. Since our model predicts global energies per atom, it gives the exact same result for a system that is simply periodically duplicated. This suggests that the predictions made when extrapolating to tiled systems that have been perturbed should maintain reasonable accuracy. This figure agrees

with this conjecture since the corresponding error lines follow a similar trajectory to the line for the small system test data. This figure also shows that simpler models are favored for the independently relaxed AIMD-generated systems (the "from scratch" systems). These systems are less likely than the tiled systems to be similar to examples from the training set. The rapid increase in error on large systems for higher model complexity illustrates the sensitivity of the task to overfitting. Nevertheless, as depicted in the left panel of Figure 4.2.2, the optimal number of features for interpolation on amorphous $Li_\alpha Si$ data is approximately the same as the optimal number of features for energy predictions on the collection of states with larger unit cells. From Table 4.1 (third row, "large states"), we see that while the prediction errors are higher for the larger systems, it is not an unreasonable increase from errors on the smaller systems.

4.2.3 Bulk Modulus

Finally, we consider elastic property data. Elastic properties are another important output of atomistic simulations. These are typically calculated by applying small strains to the system in question and fitting elastic constants to the energy-versus-strain [80]. This too is an extrapolation task for our model because uniformly expanded or compressed structures do not appear in the training set. Testing data for this task were generated based on the lowest-energy structure at each concentration by applying hydrostatic strain and varying the side-length of the simulation box from -9% to 9%.

The bulk modulus K is calculated by fitting data near the minimum to the following equation:

$$K = V_0 \frac{\partial^2 E}{\partial V^2} \tag{4.2}$$

where V is the volume, V_0 is the equilibrium volume, and E is the energy. In total, a bulk modulus value is calculated at each of the 37 concentrations, based on a total of 333 structures under hydrostatic strain.

The energy vs volume of the strained structures (Figure 4.5) shows remarkable agreement between DFT and the ML model. The RMSE curves shown in Figure 4.6 and the average errors

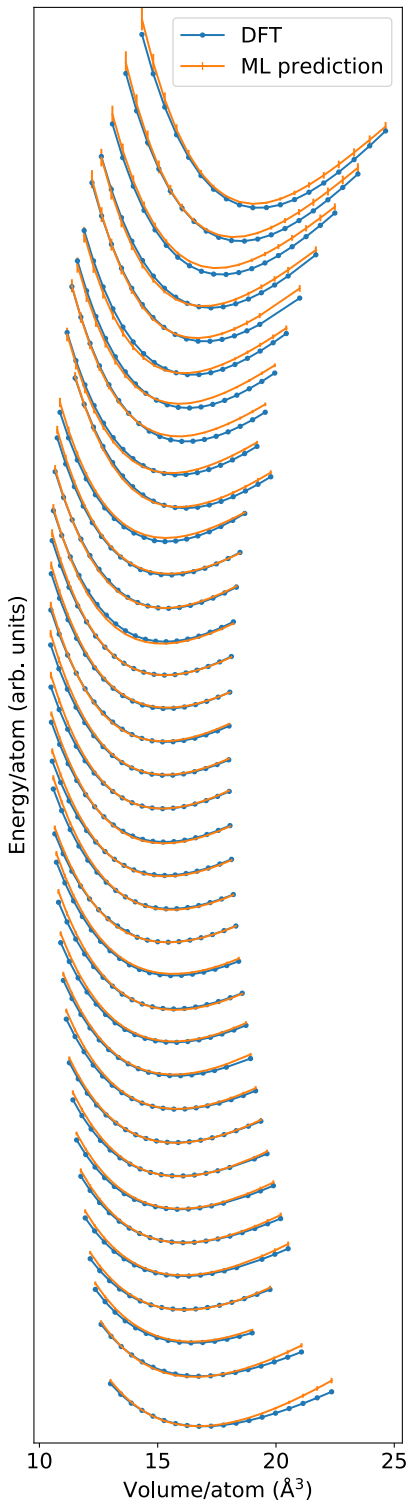

Figure 4.5: Energy per atom of hydrostatically strained $Li_\alpha Si$ structures as a function of volume per atom. Energies are shifted vertically to avoid overlap: α increases down the vertical axis. Error bars on ML prediction show the standard deviation of predictions of the the 5-fold cross-validated models for each structure.

in the last row of Table 4.1 (bulk modulus) are also quite low. The predicted bulk modulus of the structures is shown to decrease as a function of lithium content in Figure 4.6. The ML method accurately captures lithiation induced softening of the silicon. Energy-vs-strain curves along different deformation paths may also be used for the estimation of additional thermodynamic parameters, including Young's modulus, shear modulus free energies, and heat capacities through the Debye–Grüneisen model [81, 82].

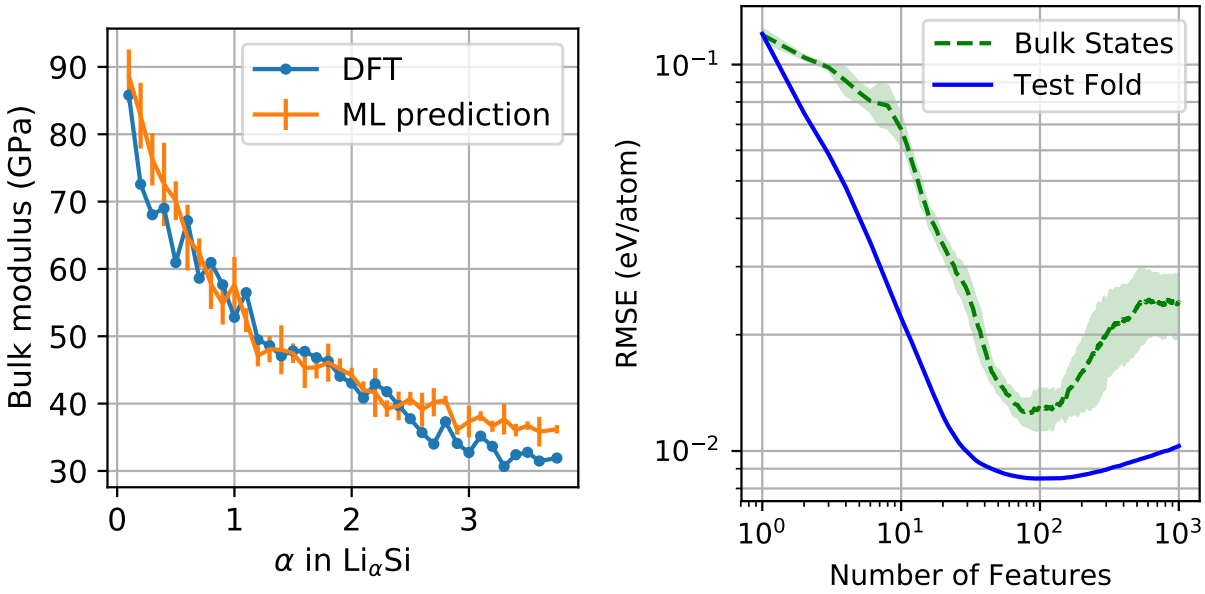

Figure 4.6: (left) Comparison of DFT-calculated bulk modulus and ML-predicted bulk modulus. Modulus was calculated through a parabolic fit to points within ± 4% strain of the energy minimum. Error bars on the ML prediction show the standard deviation of fitted modulus across the 5-fold cross-validated models. Averaging across the folds leads to a prediction with MAE of 3.3 GPa compared to the DFT values. (right) A log - log plot of average of RMSEs of models on the interpolation test set from Section 4.1 and on bulk modulus data. Here, y-axis = log(eV/atom), x-axis = log(number of features in models). Green curve is the average of the RMSEs for each fold with error bar given by the standard deviation over the five folds. As for the large states (see Figure 4.2.2), we see that the location of the minimum (i.e., the optimal number of features) for the interpolation test error is similar to the extrapolation error for the bulk modulus states.

4.3 Ablation Study of Adaptations

The model used in the previous two sections (hereafter referred to as the full model) has numerical results on the test set summarized in Table 4.1 and the training method is described in

Section 4.1. The results of two alternative models on the various tasks of this work are listed in Tables 4.3 and 4.4. The test folds of relaxation paths and $Li_\alpha Si$ states of the diffusion, large states, and bulk modulus states are identical in all three model comparisons.

The first alternative model (hereafter the 0-1 model) is trained identically to the full model with five test folds (the test folds are identical for both models) and 100 randomly selected sets of relaxation strings (with replacement) for training, but with only zero and first order wavelet scattering features available for selection in training. This results in a total of 321 features (with bias) to select from compared to 3741 in the full model. Note that at each step of the greedy OLS training the best feature is chosen from $\sqrt{321} \approx 17$ features that are randomly selected from the remaining unselected features compared to $\sqrt{3741} \approx 61$ in the full model. The model size $M^\star$ averaged over all 500 permutations of the training data is 121 in the full model and 108 for the 0-1 model, with standard deviations of 64 and 38, respectively. The numerical results for the 0-1 model are listed in Table 4.3. The performance is comparable on the relaxation paths. On the diffusion states the 0-1 model has slightly better performance in RMSE and MAE, but the standard deviation in MAE across the five folds is nearly double the full model. Furthermore, inspection of the barriers computed by the 0-1 model reveals that they are in fact slightly worse than the full model. The performance of the 0-1 model is significantly worse than the full model on the large and bulk states, again with a large spread in errors. This indicates that we get a statistically significant benefit by including second order features in the models.

	RMSE (meV/atom)	MAE (meV/atom)
Relaxation paths	8.04 ± 0.59	5.99 ± 0.39
Diffusion	11.8 ± 0.48	9.51 ± 0.95
Large states	14.0 ± 0.68	10.2 ± 0.42
Bulk modulus	39.5 ± 4.82	25.1 ± 2.92

Table 4.3: Numerical results for ML predictions with only zero and first order features (compared to zero, first, and second in Table 4.1) on the test data from the amorphous dataset and the three extrapolation tasks from the model trained only on the amorphous data.

The second alternative model (hereafter the non-randomized model) has the same features available as the full model and the same five test folds as the prior two models. The training set

is randomly partitioned into four equally sized sets (selection by relaxation strings) with a model trained for each selection of a set as validation and the remaining three for training (i.e., nested five-fold cross validation, as in [83]). This results in four trainings for each test fold for a total of 20 models trained compared to the 500 trainings (five test sets with 100 training/validation splits) of the full model. This non-randomized procedure ensures uniform representation of the strings in the training, validation, and testing folds. During training of the non-randomized model the OLS algorithm seeks the next best feature at each step from all remaining features rather than randomly selecting a subset of features to choose from as in the prior two models. The average value of $M^\star$ is 153 for the non-randomized model with standard deviation of 82 across the 20 trainings. The performance of this model is similar to the full model on relaxation paths but with significantly larger spread of the errors between models on the diffusion, large, and bulk states. Thus the chance of a catastrophic error is higher. Furthermore, the RMSE and MAE are significantly larger on the bulk states. This indicates that the model over-fit the training data and did not generalize as well to the extrapolation tasks. Randomized training in the full model appears to mitigate the possibility of over-fitting.

	RMSE (meV/atom)	MAE (meV/atom)
Relaxation paths	7.50 ± 0.39	5.64 ± 0.28
Diffusion	11.6 ± 1.01	11.0 ± 1.03
Large states	9.78 ± 1.98	6.60 ± 0.81
Bulk modulus	16.6 ± 4.91	11.5 ± 3.55

Table 4.4: Numerical results for ML predictions with the non-randomized model. The models are trained without random feature selection at each step of the greedy OLS algorithm, i.e., at each step all features are available for selection.

For our final consideration of alternative models, we look back to the model used in Chapter 3 (hereafter the random sampling model). As mentioned in Section 4.1, the random sampling model contains many more features (2855 ± 113 features) than the full model. This is due to the validation data matching the training data much more closely. This causes a problem for the generalization tasks. In Table 4.5, we can clearly see that the results are nowhere near the results with the other models. We know that the random sampling model gets excellent results on the

original lithium-silicon data set. However, since it was not created with the goal of extrapolation, we get a model which is not overfit in the traditional sense because the results on the training data extend nicely to the testing data. The difference between the full model and the random sampling model demonstrates that it is necessary to account for how the model will be used when selecting the training process. Additionally, we see that minor concessions in interpolation error can lead to significantly improved results in the extrapolation setting.

	RMSE (meV/atom)	MAE (meV/atom)
Diffusion	928 ± 4.00	928 ± 4.54
Large states	687 ± 57.2	662 ± 53.3
Bulk modulus	663 ± 11.9	674 ± 20.4

Table 4.5: The results are much worse than the other models considered for testing on the extrapolation data. This is expected due to the nature of the sampling and the way the effect that had on the model.

4.4 Conclusions

We have demonstrated a machine-learning model based on wavelet scattering that can achieve an accuracy of 5.52 meV/atom (mean absolute error) in energy on the prediction of amorphous $Li_{\alpha}Si$ structures. We have tested the generalizability of this energy predictor on three extrapolation tasks: diffusion barriers, large systems, and bulk moduli. As expected based on the nature of regression-based ML, if care is not taken to avoid over-fitting, the model performs poorly on these extrapolation tasks. However, we have shown that a statistically based feature randomization procedure, using the universal wavelet scattering features, can significantly enhance performance on the extrapolation tasks without significant reduction in performance on the interpolative test set.

Although the present work is limited to amorphous $Li_{\alpha}Si$, it provides general lessons for those wishing to apply ML models to new problems in chemical physics. This is often a daunting task because ML is generally an interpolative technique. Before a model can be used, it must be trained on large amounts of data similar to the task at hand. If the problem is new or challenging to solve by conventional means, the generation of these data can be quite difficult. Extrapolation from the well-known systems may be possible, but off-the-shelf ML models do not extrapolate

well. However, extrapolation performance can be greatly improved by taking a different approach to training the ML model.

Simpler models generalize better. In our model, "simplicity" corresponds to the number of features (wavelet scattering coefficients) used and the fact that these features provide unsupervised descriptions of atomic states, but the concept is general. Validation sets are often used in machine learning to choose a model complex enough to describe the training data but simple enough to avoid over-fitting. By utilizing randomized feature selection and the aggregation of an ensemble of models (bootstrapping), we obtain a robust and accurate model when applied to the aforementioned extrapolation tasks. From this perspective, typical ML metrics such as testing and validation error are not the only criteria for a "good" model.

In order to apply these principles to harder extrapolation tasks and to incorporate a priori uncertainty quantification, it will be necessary to leverage statistical methods that allow one to predict which properties will be difficult for the model, suggesting possibilities for efficient training set expansion to further improve generalizability. Training set expansion could be automated using "active learning," allowing a model to improve itself based on problems presented to it. The linear regression model over unsupervised nonlinear wavelet scattering features is well positioned for such future work, as it is relatively simple (compared to fully supervised neural networks) to incorporate new data on the fly.

In future work, we will extend the model to include force predictions. Our energy predictions are given as linear combinations of features that are each dependent on the atomic positions. The features are differentiable and we can carry out this differentiation analytically, which opens up fast force computations and fits the weights of our model to force data. In this case, all the methods of our model would still be applicable. This has a computational advantage over features with learned filters, which would likely use automatic differentiation. Including forces in the weight learning process could affect the weights that are learned for the energy predictions as each system would have $3N_x$ more points of training data. We expect that this will boost the generalization ability of the model. In this regard, training on forces will act as a regularizer for energy predictions.

Going beyond the basic predictions we initially made on the Lithium Silicon data base, here we demonstrate the accuracy of the model across data not seen during training. This implies that we are able to capture some true chemistry in our models.

CHAPTER 5

THE PATH ON GRAPHS

5.1 Translating Voxels to Points

We now consider an alternate approach that offers an improvement in the computation scaling with the number of atoms. With an image in 3D space, we have a sampling of many points within the box inside which the atomic structure exists. The representation scales as $O(N^2)$ which is better than most chemical methods, but less than desirable for some practitioners. As discussed in Section 2.5, it is possible to make adjustments to the representation so that the scaling is improved to $O(N(\log N)^\delta)$ for some $\delta \geq 1$ with a large hidden constant. The density of sampling allows us to characterize every point in that space. However, the points that we are specifically interested in are the atom locations. The surrounding sampling is only useful insomuch as it contributes to the richness of the representation of the atoms themselves and the interactions between atoms. We now consider a point convolution approach with the atom locations as the only positions considered [34].

Rather than a density, the point convolution approach operates on a matrix of differences of positions of the atoms. This scales as $O(N^2)$ since it contains every atom pair but it has a much lower scaling prefactor than the voxel approach, and has further opportunity for improvement. The most basic version is the matrix $D \in \mathbb{R}^{N\times N\times 3}$ containing the pairwise distance vectors with a zero diagonal.

$$D = \begin{bmatrix} 0 & R_1 - R_2 & \dots & R_1 - R_N \\ \vdots & \vdots & \ddots & \vdots \\ R_N - R_1 & R_N - R_2 & \dots & 0 \end{bmatrix}$$

Since all of the information in these matrices is relative to other atoms in the structure, D is globally translation invariant. The matrix D is rotation equivariant for each entry. If a structure is rotated, the vectors between pairs rotate correspondingly.

The distance matrix is a precise representation of the structure. In the voxel case, we have a set resolution which controls the highest frequency information that will be preserved. Since the frequency is bounded in the representation there will exist some small movement of the atoms which will be undetectable in the representation. While this error may be acceptably low, defining the atom positions as the points of interest removes any approximation.

Since the wavelets we designed for the voxel-based approach worked well to describe the lithium-silicon structures, we retain them to apply to this representation. As output we get a vector representation of the atomic structure. In this chapter we define point convolution as

$$\rho_x^c * \psi_\lambda(R_i) = \sum_{k=1}^{N} \rho_x^c(R_k)\psi_\lambda(R_i - R_k) = \sum_{k=1}^{N} c(Z_k)\psi_\lambda(D[i,k]) \tag{5.1}$$

which is the same convolution as before, but evaluated at R_i instead of u. Also, for wavelets with small support in space, we can restrict the set of atom pairs on which we evaluate the summation since distant atoms pairs would be essentially zero. This can result in $O(N)$ scaling as opposed to $O(N^2)$ scaling, but only for small wavelets.

Since each point in the input of this function depends on all the interactions with a particular atom, the output is equivariant with respect to permutations. Unlike in the voxel approach, here, we retain individual representations for each atom, but must treat them identically modulo atom type. Also, unlike the voxel formulation, the point convolution is invariant to translations, not equivariant.

The point convolutions with our wavelets are not immediately equivariant to rotations, and we must apply the special modulus as before.

$$\sigma(\rho * \psi_\lambda)(R_i) = \left(\sum_{m=-\ell}^{\ell} |\rho * \psi_\lambda(R_i)|^2 \right)^{1/2} \tag{5.2}$$

The application of the special modulus here makes the representation invariant to rotations rather than equivariant to rotations. Note again, that this is a difference from the voxel scattering representation. However, since this representation does not have a geometric underlying structure, it is not immediately clear what information it contains. In particular, it is important to discern

if angular information is preserved or if the processing of the wavelet transform in this context effectively reduces the wavelets to radial functions.

As a test, we consider a structure of three atoms. Each index in the representation contains a feature that is determined by all three atoms.

$$\rho_x^c * \psi_\lambda(R_i) = \rho_x^c(R_k)\psi_\lambda(R_i - R_k) + \rho_x^c(R_j)\psi_\lambda(R_i - R_j) \tag{5.3}$$

In this case, the central angle of each is recoverable with a dictionary of sufficiently many wavelets.

$$\exists (w_\lambda)_\lambda \text{ such that } \sum_\lambda w_\lambda \rho_x^c * \psi_\lambda(R_i) = \angle jik \tag{5.4}$$

With more than three atoms, the angular information is lost because Equation 5.3 becomes a function of the distances between at least four atoms. This demonstrates that, while it is initially accessible, angular information is collapsed almost immediately.

In the voxel approach, the entries of the intermediate representation which correspond to an interaction between two or more atoms exist at the location of the interaction on the grid. In the point convolution approach, that information is stored at the location of the central atom. This is unfortunate because the contributions to the feature value lack a directional component. It would be possible to place new nodes between atom locations to augment the geometric information, but that would be harmful to the scaling of the algorithm and would present new challenges for the implementation. At this point, the local information is significantly different from the voxel approach. In the point convolution formulation, we do not have location mappings for the contributions due to other atoms which is important to consider as we proceed into a second layer.

The scale-coupling layer will appear as

$$\begin{aligned}\sigma(\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2})(R_i) &= \left(\sum_{m=-\ell_2}^{\ell_2} |\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2}^m(R_i)|^2 \right)^{1/2} \\ &= \left(\sum_{m=-\ell_2}^{\ell_2} \left| \sum_{k=1}^{N} \sigma(\rho_x^c * \psi_{\lambda_1})(R_k)\psi_{\lambda_2}^m(R_i - R_k) \right|^2 \right)^{1/2}\end{aligned}$$

Note the difference in the flow of information. In the voxel approach, we begin at the atom centers and emit a filter, then sum over the squared filter responses to get rotation equivariance.

The interactions between atoms is encoded in the space between them through the interference of the wavelets. The second layer then combines information from everywhere within reach of the wavelet to each grid point where the convolution is evaulated. In the point convolution approach, the information regarding all neighbors (as defined by the filter support) is immediately gathered to each atom center. In the second layer, we have each neighboring atom contribute its own neighborhood's information to the central atom.

Recall that the voxel scattering, before integration, is equivariant to translations and rotations and invariant to permutations. By computing the integral of the voxel scattering maps we transformed the isometry equivariant maps into an isometry invariant representation. Here, the point convolution scattering is the opposite; it is invariant to translations and rotations but equivariant to permutations. Here, we need to collapse the permutation equivariant intermediate representations to permutation invariant features instead of collapsing isometry equivariant representations to isometry invariant features. We define the features with the same notation as before, but here instead of an integral we sum over the entries of the representation vector.

$$\|\sigma(\rho_x^c * \psi_\lambda)\|_q^q = \sum_i |\sigma(\rho_x^c * \psi_\lambda)(R_i)|^q \tag{5.5}$$

$$\|\sigma(\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2})\|_q^q = \sum_i |\sigma(\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2})(R_i)|^q \tag{5.6}$$

The resulting representation is permutation invariant.

If we continue to follow the approach from the previous chapters, we would now also use a greedy linear regression to learn weights which combine the features and make predictions of the energy. The improved scaling of the point convolution framework allows us more options for the combination of the features. Due to the lack of interference patterns between the atoms, it seems likely that these features are less expressive and need to be improved upon to get good results.

5.2 Expanding Model Flexibility

The computational efficiency of the point convolution approach allows us to place it into a neural network framework. This has some advantages. We can learn weights at several points

along the algorithm. The first location is in the point convolution itself. We can apply the selection of density channels that we developed, but $c(Z_k)$ does not need to be a predetermined function of the charge and can instead be a learned embedding of the charge. This is similar to what is done in SchNet [41]. By creating a matrix with one-hot encoding for the atom type, we can learn a weight matrix to apply to it which will yield representations which are consistent within each atom type and unique among different atom types. More specifically, let $\boldsymbol{X}_0$ be the $N_x \times d_0$ matrix, where d_0 is the number of atom types, that consists of the one-hot encoding of each atom in x. Now let $\mathbf{W}_0$ be a $d_0 \times \tilde{d}_0$ learned weight matrix. Then $\boldsymbol{X}_0\mathbf{W}_0$ defines $\tilde{d}_0$ channels of the state x.

In the scattering portion of the algorithm we can learn to combine first order representations prior to the second convolution.

$$\sigma(\sigma(\rho * \psi_{\lambda_1}) * \psi_{\lambda_2})(R_i) = \left(\sum_{m=-\ell}^{\ell} \left| \sum_{\lambda_1} w_{c,\lambda_1} (\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2}^m)(R_i) \right|^2 \right)^{1/2} \tag{5.7}$$

This can be linear as written above, but alternately, a nonlinearity can be applied and additional layers can be added. This gives us more flexibility in the pairing of wavelets between the first and second layer. It is also a computational advantage in that we can reduce the number of representations coming from the first layer.

This neural network framework can also include a learned combination of representations to yield the energy prediction in place of the greedy regression. This could be nonlinear and offers more options for combining the resulting wavelet scattering features. We consider an intermediary representation of size $|\Xi|$ and consider weights $w_\xi : \xi \in \Xi$.

$$\tilde{E}_f(x) = \sum_{\xi} w_\xi \mathrm{ReLU} \sum_{c,\lambda,q} w_{c,\lambda,q}^{\xi} ||\sigma(\sigma(\rho_x^c * \psi_{\lambda_1}) * \psi_{\lambda_2})||_q^q \tag{5.8}$$

In this example we first apply a learned weight matrix which collapses the features to size Ξ. Then we apply a nonlinearity followed by another weight matrix which collapses the features down to a scalar to make the energy prediction.

Let us combine all of the previous considerations and examine this algorithm in matrix form for clarity. Let $\boldsymbol{X}_b \in N_x \times d_b$ be the representation of x with d_b channels at layer b. This is a matrix

of representations for each atom in the system. We have the matrix $\boldsymbol{X}_0 \in N_x \times d_0$ which is the 1-hot encodings of atom types. This allows similar functionality but more flexibility compared to the density channels that we developed in previous sections. Define K to be the number of wavelets that we use. Let $\boldsymbol{\Psi}_x \in K \times N_x \times N_x$ be the tensor encoding wavelet point convolution operators. This is Equation (5.1) acting on the matrix D across the full dictionary of wavelets except without the channel function since we learn that separately. Let $\boldsymbol{W}_b \in d_b \times \tilde{d}_b$ be the learned weight matrix which is independent of x. The matrix $\boldsymbol{W}_b$ combines channels to form new channels. We get an intermediary matrix

$$\tilde{\boldsymbol{X}}_b = \boldsymbol{\Psi}_x \boldsymbol{X}_b \boldsymbol{W}_b \in \mathbb{R}^{K \times N_x \times \tilde{d}_b} \tag{5.9}$$

We take the special modulus of this matrix to get

$$\sigma(\tilde{\boldsymbol{X}}_b) = \sigma(\boldsymbol{\Psi}_x \boldsymbol{X}_b \boldsymbol{W}_b) \in \mathbb{R}^{\tilde{K} \times N_x \times \tilde{d}_b} \tag{5.10}$$

where $\tilde{K}$ is the number of (n, ℓ, j) combinations we consider. We reshape the matrix so that we have a two dimensional matrix

$$\boldsymbol{X}_{b+1} = \text{Reshape}[\sigma(\tilde{\boldsymbol{X}}_b)] \in N_x \times \tilde{K}\tilde{d}_{b+1} \tag{5.11}$$

We then define $d_{b+1} := \tilde{K}\tilde{d}_{b+1}$. We can repeat this process for B layers leading to $\boldsymbol{X}_B \in N_x \times d_B$. Each layer refines the features of each atom and propagates them to the other atoms. We apply the discrete q-norm to the N_x dimension

$$\phi_\gamma(x) = \sum_{i=1}^{N_x} |\boldsymbol{X}_B[R_i, k]|^q, \quad \gamma = (k, q), \quad 1 \leq k \leq d_B, \quad q \in \{1, \frac{4}{3}, \frac{5}{3}, 2\} \tag{5.12}$$

which gives the representation

$$\Phi(x) = (\phi_\gamma(x))_{\gamma \in \Gamma}, \quad |\Gamma| = 4d_B \tag{5.13}$$

Then we also learn regression weights (in tandem with the weight matrices $(\boldsymbol{W}_b)_b$, $0 \leq b < B$).

$$\tilde{E}_1(x) = \sum_{\gamma \in \Gamma} w_\gamma \phi_\gamma(x) \tag{5.14}$$

Alternatively we can learn to combine the last layer of the representation with an artificial neural network

$$\tilde{E}_1(x) = \text{ANN}(\Phi(x)) \tag{5.15}$$

5.3 Training with Forces

We consider a different data set to test these methods. MD17 is a data set of molecular dynamics trajectories on several small molecules (aspirin, benzene, ethanol, malonaldehyde, naphthalene, salicylic acid, toluene, and uracil) [24, 84, 85]. The number of data points ranges from 150k to nearly 1M for each molecule. The data includes energy in kcal/mol as well as forces in kcal/mol/Angstrom. This data set has been used as a benchmark for many papers in this field [17, 41, 86, 87, 88, 89].

Initially many researchers trained on a selection of 50k points per molecule. These points are selected randomly from among all conformational geometries. The data is different from our lithium silicon data in that there is a single trajectory for each molecule. Despite the time series nature of this data, the standard practice for selecting training data does not take into consideration the data generation. With only one simulation to sample from, sampling to the first 50k geometries might be too restrictive to be able to predict the rest of the data points in the simulation.

In more recent years, researchers have reduced the training set size to only 1000 points. The dramatic reduction is balanced through the use of forces in the training process. Forces offer significant improvements in reducing the prediction errors on energy. In standard machine learning, second-order gradient methods are desirable for their superior performance, but are typically not used because of their computational cost. Methods like ADAM were introduced as a cheaper substitute [3]. For machine learning in quantum chemistry, the benefits of computing a second derivative are even greater because we are not trying to control the derivative of the gradient, but instead we are trying to match the gradient to specific values.

$$p \sum_{i=1}^{n_t} |E_1(x_i) - \tilde{E}_1(x_i)|^2 + (1-p) \sum_{i=1}^{n_t} \frac{1}{N_{x_i}} \left\| \nabla E_1(x_i) - \nabla \tilde{E}_1(x_i) \right\|^2 \tag{5.16}$$

For an energy prediction $\tilde{E}_1$ we can reformulate the optimization as minimizing a new loss (5.16) with added hyperparameter $0 \leq p \leq 1$ which controls the balance between energy and forces.

At first we can see that minimizing (5.16) will regularize $\tilde{E}_1$. Each training point which originally had a single target will have $3N + 1$ targets. Figure 5.1 gives an example of the benefit. The resulting model will be more stable to perturbations. If the movement of a single atom had too strong of an effect on the prediction, its corresponding force prediction would be too high. Models with low losses will have to have smooth energy predictions. For the wavelet scattering model, inclusion of the forces does not have benefits beyond regularization. With the representation completely defined sans the linear combination, the model does not have much flexibility to learn from the forces. However, when the model has learnable layers throughout, the forces can contribute to guiding the function to be more chemically accurate. One advantage of the wavelet scattering method is that we can define the derivative analytically, but even so it compounds the prefactor on the scaling which is already a bit high.

The addition of forces clearly adds computation in the training phase. Naively we multiply the computational cost by $3N + 1$. However, much of the computation overlaps. In the neural network approach, we already compute the forces as the gradient of the energy. We add the second derivative which we can compute once through the entire network and then multiply by the derivative of the first layer with respect to the position once for each atom. This makes the additional computational cost bearable.

At this point, the point convolution representation that we have developed is not very rich even with the learned layers. With the rapid collapse of angular information and lack of geometric flexibility, much of what made the wavelet scattering representation good has been lost. When training this model (without forces) on the molecules in MD17, the results for seven of the eight are not much better than predicting the mean for every data point. See Table 6.1. Our predictions for benzene are very near to state-of-the-art. We present results of our representations' predictions on this data set in the following chapter. The results from the model in this section suggest that for extremely symmetric structures, this representation can capture the perturbations of the atoms.

To get better results, we must add back information that the voxel wavelet scattering captured, in a way that can be applied to the point convolution setting.

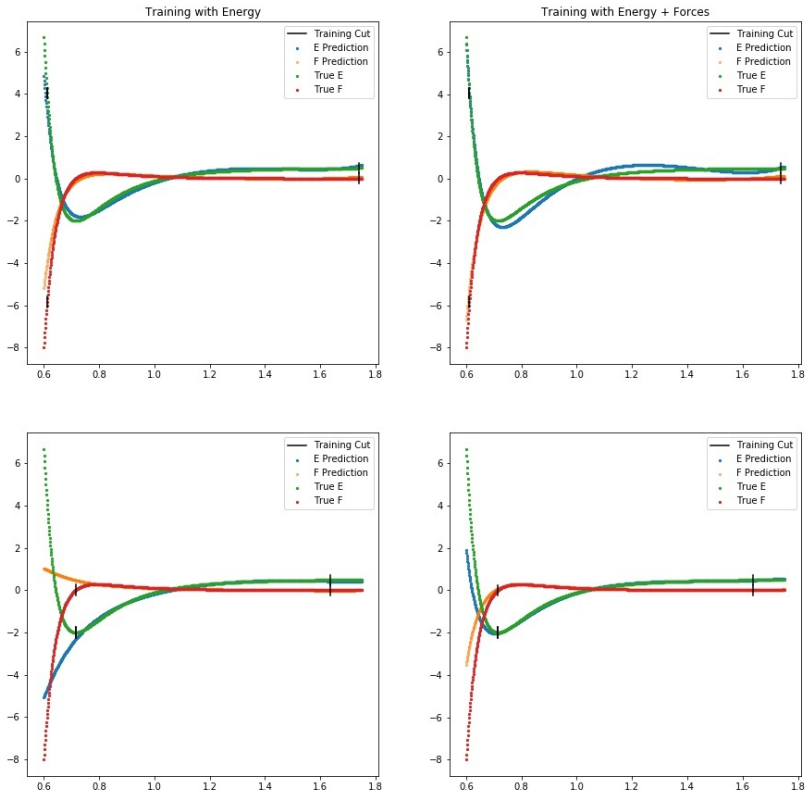

Figure 5.1: A toy example of the benefits of training with forces. For a two-atom system we consider energy and force derived from the Lennard-Jones potential. The y-axis measures energy and forces. The x-axis describes distance between the two atoms. As the input to a three layer fully-connected network, we take powers of the inverse distance: $(|R_1 - R_2|^{-p})_p : p \in [1, \dots 9]$. The training cuts in the plots indicate the endpoints for the training sampling. In the top left, we get good results for both energy and forces when training on only energy over the full energy surface. However, in the bottom left, we see that when the extreme energy regions of close distances are excluded, the predictions are drastically worse. In the bottom right, we notice that with the inclusion of forces, we are able to predict the close distance energies fairly accurately without any training data from that region.

CHAPTER 6

AUGMENTING THE REPRESENTATION

Inspired by molecular dynamics simulators, the energy of an atomic structure can be decomposed into a sum over five parts [90].

$$E = E_{bonds} + E_{angle} + E_{torsion} + E_{electrostatic} + E_{vanderWaals} \tag{6.1}$$

In this equation, we have several different types of interactions represented. The first term encodes pairwise interactions between nearby atoms. The second term captures three-body interactions between neighboring atoms. The third term depends on the dihedral angles between the planes of two pairs of bonded atoms. The fourth and fifth terms model the interactions between non-bonded atoms which are farther apart. This decomposition of the energy is common in the context of force fields. We consider it here to assess important elements for predicting the ground state energy of molecules.

The wavelet scattering approach includes information from each of these terms in the voxel representation. There are bonded and non-bonded terms depending on the scale of wavelet used, but there is no direct separation of the representations into the terms in (6.1) (i.e. the large wavelets still capture local interactions), and angles and torsion are implicit. With a point convolution approach, we have flexibility to expand the representation to include these energy terms more directly.

Additionally, we can improve the scaling of the algorithm from the $O(N\log(N)^{\delta})$ scaling of the wavelet scattering approach. One can achieve $O(N)$ scaling when considering only the local neighborhoods of each atom. This has been the primary approach for graph-based methods in the field of machine learning for quantum chemistry [42, 41]. For message-passing based models [31], information is propagated between edges of the molecular graph. In this way, the information between distant atoms eventually interacts, but it has been diluted as it passes through each atom in between them to the point that it is challenging to get an accurate representation of the long range non-bonded interactions.

We would like a representation which encodes different scales explicitly. The best scaling interaction scheme between atoms which contains information over multiple scales in a molecule is $O(N)$. This is achieved by looking at only local interactions and then aggregating them by group to form global representations. The fast multipole method is a computational technique that has been widely adopted using this type of aggregation scheme over scales [91]. This is also similar to what happens for a two-layer wavelet scattering representation. For the wavelet scattering approach, the small filters are defined on a grid that is the same size as the large filters so there is no computational savings. By stacking a small filter in the first layer with a large filter in the second layer, we encode the coupling of scales. We can apply the same filters in the point convolution approach, but it will not work the same. This is because the model is expressed only on atom locations and does not have the ability to aggregate at a new location that represents atom neighborhoods. Additionally, the point convolution approach applied with large filters scales as $O(N^2)$.

We now look to improve the information content of the point convolution approach. The proposed considerations will also allow us to improve computation. We are committed to developing a representation with $O(N)$ scaling, meaning that it can only consider atom neighborhoods instead of all pairwise interactions. We propose three modifications to the learned point convolution scattering introduced in Chapter 5.

1. We use only small filters (e.g., $j = 0$ only), which reduces the computational cost of a single layer to $O(N)$ instead of $O(N^2)$.

2. Using only small filters means only local information is encoded. Even by stacking layers to go from local to global as in the message-passing framework [31], information is still lost relative to a fully multiscale approach [42]. We therefore add in local 3-body angle terms to the existing 2-body point convolution terms.

3. Even with the 3-body terms, the network is still missing higher body terms. Furthermore, in order for the network to encode long range interactions, its depth would need to be equal to the diameter of the molecular graph which in the worst case scales as $O(N)$, and hence the

overall cost is still $O(N^2)$. In order to address both issues, we incorporate pooling into the point-convolution scattering network using the DiffPool framework [92]. As we illustrate, pooling between layers can transform many-body interactions into 2 and 3-body interactions. Furthermore, if one pools on the order of half the previous layer's atoms, then the depth of the network will be $O(\log N)$ and its computational complexity for evaluating a new molecule will be $O(N)$.

6.1 Inclusion of Angles

While we desire global rotation invariance, the local rotation information is important to represent bond angles well. This is included implicitly in the voxel approach through the wavelet interference patterns between atom locations, but it is mostly lost in the pairwise graph approach. Using D, instead of pairwise distances, we retain the orientation of the structure and compute the angle $\angle ijk$ from the vectors R_{ji} and R_{ik}. We can create a three-body representation as

$$\rho_x \diamond Y_\ell^m(R_i, R_j, R_k) = Y_\ell^m(R_i - R_j) + Y_\ell^m(R_i - R_k) \tag{6.2}$$

We leave out the channels for now and will reintroduce them to the representation later on. This representation is globally rotation invariant, but can recover the angle $\angle_{jik}$ with a large enough set of filters (large L for $0 \leq \ell \leq L$) indicating that it retains local rotation information. The issue with this representation is that for large ℓ, there are many m's to go over. An alternative approach is to compute the angle explicitly and compute the spherical harmonic at $m = 0$ for various ℓ values. This set of spherical harmonics have a term cancelled out which reduces them to the Legendre polynomials P_ℓ with a constant in front. These polynomials take the cosine of a single angle as their argument. This approach reduces the computation but still forms an angular basis. Let us redefine the diamond operator.

$$\rho_x \diamond P_\ell(R_i, R_j, R_k) = P_\ell(\cos(\angle_{jik})) + P_\ell(\cos(\angle_{kij})) \tag{6.3}$$

The number of terms in the first formulation is $N_h(L+1)^2$ for N_h atoms in the neighborhood. In the second formulation there are L $\binom{N_h}{2}$ terms. If we take $L = 5$ as an example, then the

second formulation is computationally superior for any $N_h < 73$. Since we are considering local representations, we will never reach 73 atoms in a neighborhood. For this reason, we discontinue computing over the entire set of m's going forward and consider representations based on (6.3).

With a general notion of how we will approach the angular representation, we turn to the radial representation. Within the neural network setting, rather than defining a set of radial functions, we can learn them. When we defined the wavelets, we created them as complete filters comprised of angular and radial terms. However, when learning filters, the radial portion can be divorced from the angular portion of the representation. In the end, we want these terms to be united to form a complete representation of the neighborhood. If we keep them together, we have representations of the form

$$\rho_x \diamond F_{\gamma\ell}(R_i, R_j, R_k) = Q_\gamma(|R_i - R_k|)P_\ell\cos(\angle_{ijk})) \tag{6.4}$$

where $F_{\gamma\ell} = (Q_\gamma, P_\ell)$ and $\angle_{ijk}$ is the angle centered at R_i and defined by R_j and R_k. Allowing the radial and angular functions to act independently of one another we have access to functions that combine various radial representations and various angular representation each before merging the two. This can be linear as described below, but the learned weights could also be expanded to multiple fully-connected layers with a nonlinearity.

$$\rho_x \diamond F^\theta_{\gamma\ell}(R_i, R_j, R_k) = \sum_\gamma \sum_\ell \theta_{\gamma\ell} Q_\gamma(|R_i - R_k|)P_\ell\cos(\angle_{ijk})) \tag{6.5}$$

While we can select and design filters that seem well-suited for the task and perhaps achieve success, having more flexibility for the model allows a better minimal error with the drawback that it requires a more challenging optimization.

To create a simple model with much flexibility, we further simplify the representation and fix the diamond notation.

$$\rho_x \diamond F^\theta_{\gamma\ell}(R_i, R_j, R_k) = \sum_{\gamma_1,\gamma_2} \theta_{\gamma_1\gamma_2}(G_{\gamma_1}(|R_i - R_k|)G_{\gamma_2}(\cos(\angle_{ijk}))) \tag{6.6}$$

where G_γ is a Gaussian basis.

$$G_\gamma(r) = \exp^{-\frac{(r-\gamma)^2}{2\sigma^2}} \tag{6.7}$$

For the radial basis, G_{γ_1} is a set of Gaussian functions with γ_1 ranging from 0 to our cutoff distance. The cutoff distance is a hyperparameter dictating the largest size of the atomic neighborhood. The standard deviation is set so that the Gaussians overlap at one standard deviation from their mean. For the angular basis we also use Gaussians but with γ_2 varying from -1 to 1 since they will act on the cosine of the angle. These bases partition the interactions of the atoms based on their distances and angles smoothly since the Gaussians overlap.

The set of representations that we get are filters that vary radially as well as angularly. A selection of the learned filters can be seen in Figure 6.1. They are two-dimensional and symmetric across the y-axis. The representations are invariant with respect to rotations because the filters effectively rotate with the atoms.

Let us once more turn to matrix notation to describe this representation more clearly. We have a matrix $\boldsymbol{G}_{\Gamma_1} \in |\Gamma_1| \times N_x \times N_x$ which is a Gaussian basis of size $|\Gamma_1|$ operating on pairwise distances of the atoms $|R_i - R_k|$ and a matrix $\boldsymbol{G}_{\Gamma_2} \in |\Gamma_2| \times N_x \times N_x \times N_x$ which is a Gaussian basis of size $|\Gamma_2|$ operating on the cosine of the angle centered at the atom in the second index $\cos(\angle_{ijk})$. We tile $\boldsymbol{G}_{\gamma_1}$ to create a fourth dimension of size N_x and take the outer product of these two representations.

$$\boldsymbol{G}_{\Gamma_1} \otimes \boldsymbol{G}_{\Gamma_2} \in \mathbb{R}^{|\Gamma_1| \times N_x \times N_x \times N_x \times |\Gamma_2|} \tag{6.8}$$

which has entries

$$\boldsymbol{G}_{\gamma_1}(|R_i - R_k)|)\boldsymbol{G}_{\gamma_2}(\cos(\angle_{ijk})), \quad \gamma_1 \in \Gamma_1, \quad \gamma_2 \in \Gamma_2, \quad 1 \leq i, j, k \leq N_x \tag{6.9}$$

and reshape to $\mathbb{R}^{|\Gamma_1| \cdot |\Gamma_2| \times N_x \times N_x \times N_x}$ before applying a learned weight matrix $\tilde{\Theta}_b \in \mathbb{R}^{\tilde{d}_b \times |\Gamma_1| \cdot |\Gamma_2|}$ to get

$$\tilde{\boldsymbol{F}}_b = \tilde{\Theta}_b \text{Reshape}(\boldsymbol{G}_{\gamma_1} \otimes \boldsymbol{G}_{\gamma_2}) \in \mathbb{R}^{\tilde{d}_b \times N_x \times N_x \times N_x} \tag{6.10}$$

A single entry of $\tilde{\boldsymbol{F}}_b$ is $\rho_x \diamond F^{\theta}_{\gamma_1\gamma_2}(R_i, R_j, R_k)$. To create filters which are nonlinear combinations of the filters we have defined, we apply an activation function. We use the shifted softplus following SchNet: $ssp(\tilde{\boldsymbol{F}}) = \log(0.5 \exp^{\tilde{\boldsymbol{F}}} + 0.5)$ and then apply a second learned weight matrix $\Theta_b \in \mathbb{R}^{d_b \times \tilde{d}_b}$

$$\boldsymbol{F} = \Theta_b ssp(\tilde{\boldsymbol{F}}) \in \mathbb{R}^{d_b \times N_x \times N_x \times N_x} \tag{6.11}$$

The matrix $\boldsymbol{F}$ can be viewed as the evaluations of a set of learned 2D filters on $|R_i - R_k|$ with an orientation governed by the vector R_{ij}.

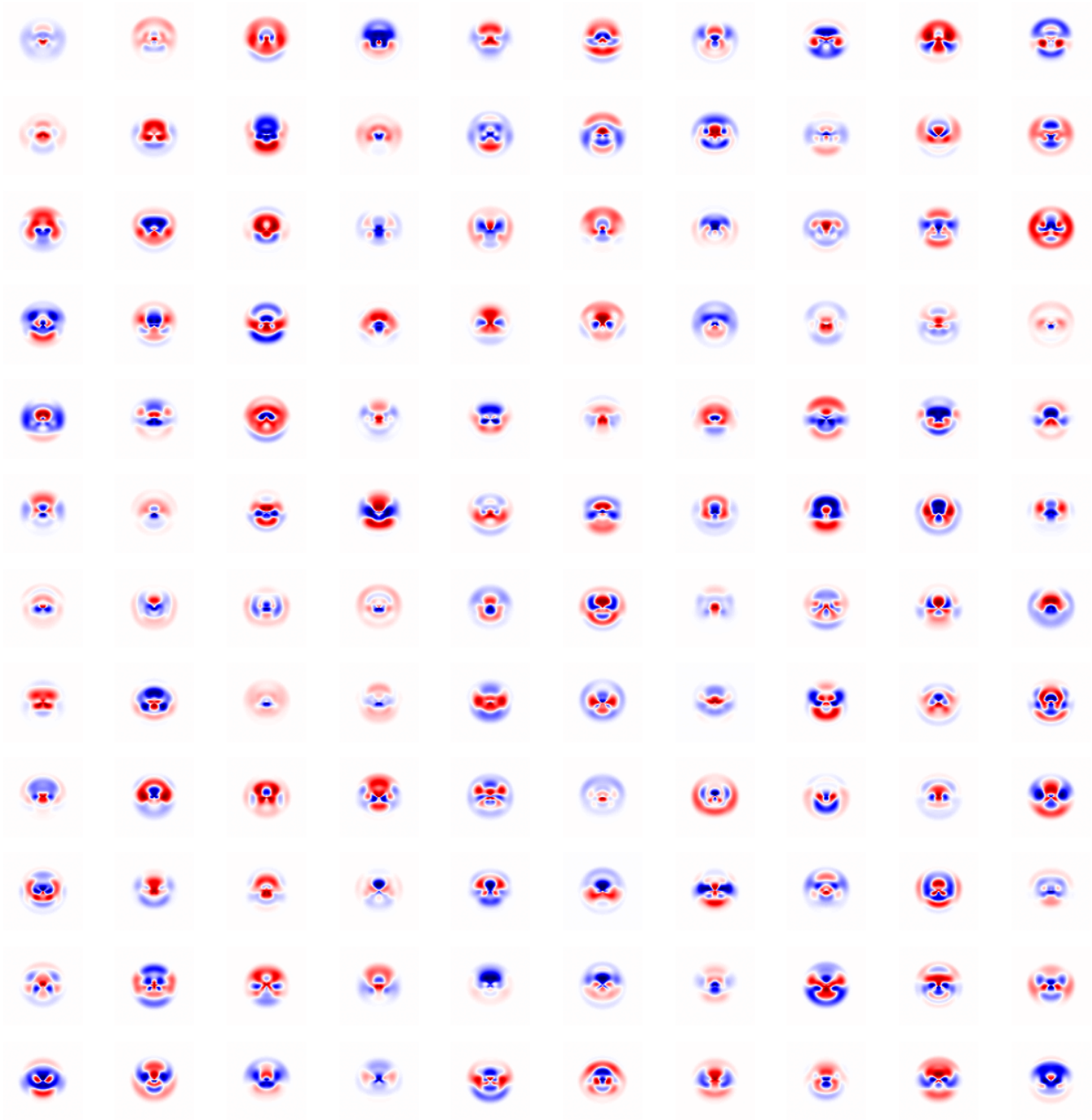

Figure 6.1: Filters learned for processing ethanol. The top third are from the first layer, the middle third are from the second layer, and the bottom third are from the third layer. For an intuitive ideal of how these are applied, consider R_i to be at the center of the image and R_j to be just above R_i. The color displays the filter response as a function of the location of R_k.

As in Section 5.2 we will refine this representation through multiple layers. At layer b, we

have the atom-wise representation $\boldsymbol{X}_b \in \mathbb{R}^{N_x \times d_b}$ and the matrix of three-body filter responses $\boldsymbol{F}_b \in \mathbb{R}^{d_b \times N_x \times N_x \times N_x}$ defined as

$$\boldsymbol{F}_b = \Theta_b ssp(\tilde{\Theta}_b \text{Reshape}(\boldsymbol{G}_{\gamma_1} \otimes \boldsymbol{G}_{\gamma_2})) \tag{6.12}$$

We take the element-wise product of these matrices.

$$\bar{\boldsymbol{X}}_b(\cdot, \cdot, j, k) = \boldsymbol{X}_b^T \odot \boldsymbol{F}_b(\cdot, \cdot, j, k) \tag{6.13}$$

We sum over the neighboring atoms since we need $\boldsymbol{X}_{b+1}$ to only have one atom index.

$$\tilde{\boldsymbol{X}}_b = \sum_{j,k=1}^{N_x} \bar{\boldsymbol{X}}_b(\cdot, \cdot, j, k) \tag{6.14}$$

Then we refine the atoms features further with two learned weight matrices and a shifted softplus and add this to $\boldsymbol{X}_b$ in the style of a residual network [93].

$$\boldsymbol{X}_{b+1} = \boldsymbol{X}_b + \bar{\Theta}_b ssp(\hat{\Theta}_b \tilde{\boldsymbol{X}}_b) \tag{6.15}$$

where $\hat{\Theta}_b \in \mathbb{R}^{\bar{d}_b \times d_b}$ and $\bar{\Theta}_b \in \mathbb{R}^{d|b \times \bar{d}_b}$. At the end of the final layer B we take the sum over the atoms and learned a weighted combination of the features to make the energy prediction.

$$\tilde{E}_1(x) = \Theta_B \sum_{i=1}^{N_x} \boldsymbol{X}_B \tag{6.16}$$

with $\Theta_B \in \mathbb{R}^{d_B}$.

We test this algorithm on MD17 using a basis ten functions for the angles and ten functions for the pairwise distances. The outer product of these leads to 100 unique basis filters which the model learns to combine. A subset of these combinations are shown in Figure 6.1. We set the number of layers at 3 and $d_b = 128$. We train on 1000 randomly sampled molecules for each of the 8 molecule types in MD17, validate on 100, and test on the remainder. We weight the training loss of form (5.16) to have a contribution of 0.99 from the forces and 0.01 from the energy. Our representation matches the current state-of-the-art for the force predictions on ethanol, malonaldehyde, naphthalene, salicylic acid, toluene, and uracil. The model that we evaluate the test set on is selected as the set of weights which minimizes the validation error.

The most similar model to ours that we show in Table 6.1 is DimeNet [42]. We have better results on forces for every molecule besides aspirin. We match DimeNet for predictions on energy aside from aspirin, benzene, and malonaldehyde where their predictions are better. Our model is fourth in average MAE for forces across the eight molecules. The three models that are superior are ACE [89], FCHL [87], and PaiNN [88]. ACE is a linear atomic cluster expansion that constructs bases up to four-body terms and designs a density trick to get rotational invariance. FCHL is a representation based on histograms of the radial distribution of atoms and Fourier terms describing the angular distribution of the atoms. PaiNN is an extension of SchNet which incorporates three-body terms. In the following section, we consider an additional element which may be able to further improve our results on MD17.

		SchNet	DimeNet	FCHL	PaiNN	ACE	Base	Angular
Aspirin	Energy	0.37	0.20	**0.14**	0.16	**0.14**	1.20	0.24
	Forces	1.35	0.50	0.49	**0.37**	0.42		0.59
Benzene	Energy	0.08	0.08	0.01		**0.00**	0.02	0.11
	Forces	0.31	0.19	0.06		**0.01**		0.10
Ethanol	Energy	0.08	0.06	**0.02**	0.06	0.03	0.40	0.05
	Forces	0.39	0.23	**0.14**	0.23	0.17		**0.14**
Malonaldehyde	Energy	0.13	0.10	**0.03**	0.09	0.04	1.75	0.22
	Forces	0.66	0.38	**0.24**	0.32	0.26		0.29
Naphthalene	Energy	0.16	0.12	0.03	0.12	**0.02**	0.20	0.13
	Forces	0.58	0.22	0.15	0.08	**0.12**		0.14
Salicylic Acid	Energy	0.20	0.13	**0.04**	0.11	**0.04**	1.38	0.13
	Forces	0.85	0.37	**0.22**	0.21	**0.22**		0.23
Toluene	Energy	0.12	0.10	0.04	0.10	**0.03**	1.40	0.10
	Forces	0.57	0.22	0.20	**0.10**	0.15		0.13
Uracil	Energy	0.14	0.12	**0.01**	0.10	0.03	1.43	0.11
	Forces	0.56	0.30	**0.10**	0.14	0.15		0.14

Table 6.1: We compare seven methods across the molecules of MD17. SchNet was the earliest application of a neural network on this data and was later adapted to PaiNN. DimeNet was the first work to include angles explicitly within the neural network framework and test on MD17. ACE and FCHL are the state-of-the-art at the time of writing.

6.2 The Coupling of Scales via Pooling

The message-passing framework or variations of it are popular as a basis for representations of atomic structures. However, this approach is deficient in the encoding of long-range interactions. This is documented in several papers [94, 95, 96]. If we consider alternative methods of aggregation across scales, we may be able to encode long range interactions better. We look at pooling for graphs. By allowing a graph to pool, we enable it to create new nodes which may be more accurate representations for the location of interaction information. The new nodes act as neighborhood features for the atoms averaging over the features and geometry of a set of local atoms. Graph pooling allows us to move towards the wavelet scattering representation by creating a path for the refinement of the geometric information of the atomic structure.

6.2.1 DiffPool

We consider the design of DiffPool which equips graph neural network (GNN) models with a differentiable module to hierarchically pool graph nodes [92]. We temporarily redefine variables for only this section. In the DiffPool framework, one has two matrices: the graph adjacency matrix A and the feature matrix $X \in \mathbb{R}^{N\times d}$. Using these two, one computes subsequent features using a GNN module which leads to a new matrix $Z \in \mathbb{R}^{N\times d'}$. Simultaneously, a pooling matrix $P \in \mathbb{R}^{N\times N'}$ is learned where N' is the number of nodes for the next layer. Applying the softmax row-wise limits the contribution of each old node to the set of new nodes, $S = \text{softmax}(P)$. We apply the matrix S as

$$X' = S^T Z \in \mathbb{R}^{N'\text{x}d'} \tag{6.17}$$

$$A' = S^T AS \in \mathbb{R}^{N'\text{x}N'} \tag{6.18}$$

From this one has a new set of nodes which represent aggregations of nodes from the previous layer. This new set remains permutation equivariant, but each original index now contributes to multiple new indices. The process can be repeated until either one node remains or it can be concluded by a summation over nodes as is done in the message-passing framework.

There are several complications that remain. The sequence for node size needs to be selected as a hyperparameter. This may depend on the nature of the graph data and could significantly affect the efficacy of the model. Additionally, Ying et al found that it was necessary to introduce several loss terms to coerce the pooling towards good behavior. The first additional loss term is the Frobenius norm $\|A_b - S_b S_b^T\|_F$. This encourages nearby nodes in layer b to contribute to nearby nodes in layer $b+1$. The second term is the entropy function over the rows of S. This helps each node to contribute to few nodes in the next layer. We take the base that Ying et al have developed and adapt it for application to graphs on atomic structures.

6.2.2 Pooling in the Context of Atoms

One of the main differences that we have to handle from the design of DiffPool is that our data is not truly a graph. Rather than an adjacency matrix, we have D which is nonzero for every non-diagonal entry. Additionally, our data is defined in $\mathbb{R}^3$ giving us greater geometric definition. Taking our matrix of learned atom-centered representations $\boldsymbol{X}_b$, we learn a matrix $\boldsymbol{P}$ as in Section 6.2.1. Applying the softmax, we get $\boldsymbol{S}$. We apply this matrix as in (6.17) to get $\boldsymbol{X}_{b+1}$. However, rather than applying $\boldsymbol{S}$ to an adjacency matrix, we apply it to the position matrix $\boldsymbol{R} \in N_x \times 3$ which results in a new position matrix $\boldsymbol{R}' \in N_x' \times 3$.

$$\boldsymbol{R}' = \boldsymbol{S}^T \boldsymbol{R} \tag{6.19}$$

With a set of refined positions, we can get a clear understanding of how the atoms contribute to create new nodes.

This brings us to another point. The aggregation scheme here is similar to the aggregation across scales in the wavelet scattering approach. The nodes in the second layer of the network represent neighborhoods of atoms. Learning new features from the geometry of those nodes coupled with the aggregation of features from the previous layer is similar to the second layer in the wavelet scattering model. An advantage here is the lower computational cost which allows us to go beyond two layers easily. If we collapse the nodes at layer b to half as many nodes at layer $b+1$, then the

computational cost at each layer b is $O(\frac{N_x}{2^b})$. The summation over the layers resulting in a scaling of $O(N_x)$. This means that we are creating a global representation while scaling linearly with the number of atoms in the structure.

Now let us consider an example of the advantages to this approach on the meso-scale. As mentioned in Equation (6.1), there is a torsion term which we have not discussed. To include these energy contributions, we must include two components: pooling and angles. While the intuition behind the inclusion of these terms is clear, we will now show how the combination of them can aid our model to represent true chemistry. Torsion is a four-body term which makes it challenging to represent in a computationally tractable way. This is possible if one explicitly defines the bonds between atoms, however that automatically restricts the conformational space to which the model is applicable. Our model avoids explicit bonds, but still has the flexibility to capture torsion without forcing the inclusion of every four-body term.

Take a molecule or atomic neighborhood with four atoms: R. We begin with four nodes in layer b. Rather than making a reduction of half the nodes, we have the flexibility to set layer $b+1$ to contain three nodes. Any matrix $\boldsymbol{S}$ is possible for the model to learn given that the rows each sum to 1. We consider

$$S = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1.0 \end{bmatrix} \tag{6.20}$$

The first node at layer $b+1$ is the average of the first two nodes at layer b. The second node at layer $b+1$ is the average of the second and third nodes at layer b. The third node at layer $b+1$ is the average of the third and fourth nodes at layer b. This can be seen in Figure 6.2.

Now consider a representation acting on these three nodes. This representation will include an angular term which acts upon the angle centered at the second node. This angle is a dihedral angle which for bonded atoms is the torsion angle. We cannot guarantee that our model will learn this way. In fact, it will likely not learn the precise weights necessary for the representation to process

the torsion angles. But, the model has the flexibility to do so, and beyond this, it can collapse even higher order terms down to two or three node representations.

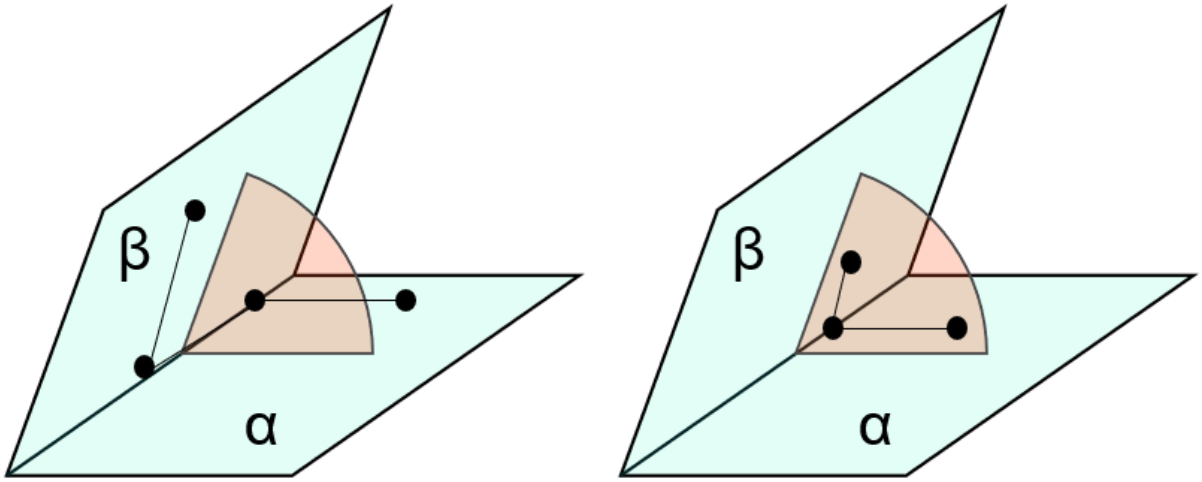

Figure 6.2: On the left, we see atoms forming two planes: α and β. The angle between these planes is the torsion angle. If the atoms are pooled to the positions on the right, then the angle between the new nodes will be a three-body representation of the torsion angle.

6.3 Conclusion

In the creation of this framework, we have developed the first machine learning representation for atomic structures which has the ability to contain explicitly the four terms of the energy decomposition in (6.1) and scales linearly with the number of atoms. We began by adapting a wavelet scattering representation into a message-passing approach. This gives us $O(N_x)$ scaling, but at the cost of the representation quality. By adding local three-body terms, we include angle information without degrading the scaling. Then by applying a graph pooling algorithm, we expanding the representation to include many-body terms while avoiding the poor scaling of a combinatorial selection over the atoms.

Several challenges remain. With the graph pooling approach, it is unclear how well hyperparameters will translate between atomic systems, particularly if they are of significantly different sizes. Since we do not explicitly define a nice representation as in the wavelet scattering approach, we depend on the optimization of the weights. There are certainly advantages to the wavelet scattering

approach over a graph neural network approach for the representation of atomic structures. Here we have shown that through careful consideration, some of those advantages can be introduced to a model that operates in the lightweight nature of a graph.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2016.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[3] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, San Diego, CA, USA, 2015.

[4] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o($1/k^2$). *Dokl. akad. nauk SSSr*, 269:373–398, 1983.

[5] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, 2013.

[6] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[7] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[8] Ding-Xuan Zhou. Universality of deep convolutional neural networks, 2018.

[9] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks, 2018.

[10] Anastasis Kratsios and Eugene Bilokopytov. Non-euclidean universal approximation. In *NeurIPS*, 2020.

[11] Wilhelm Burger and Mark James. Burge. *Principles of digital image processing*. Springer, 2013.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] K. . Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. Neur. Netw.*, 12(2):181–201, March 2001.

[16] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.*, 104(13):136403(4), 2010.

[17] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.*, 108(5):058301, January 2012.

[18] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Phys. Rev. B*, 87(18):184115(16), May 2013.

[19] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New J. Phys.*, 15:095003, 2013.

[20] Sandip De, Albert P. Bartok, Gabor Csanyi, and Michele Ceriotti. Comparing molecules and solids across structural and alchemical space. *Phys. Chem. Chem. Phys.*, 18(20):13754–13769, 2016.

[21] Alexander Shapeev. Moment tensor potentials: A class of systematically improvable interatomic potentials. *MMS*, 14(3):1153–1173, 2016.

[22] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.*, 3(5):e1603015, 2017.

[23] Felix Brockherde, Leslie Vogt, Li Li, Mark E Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the Kohn-Sham equations with machine learning. *Nat. Commun.*, 8: , 2017.

[24] Stefan Chmiela, Huziel E. Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.*, 9(1):3887, 2018.

[25] Tristan Bereau, Robert A. DiStasio, Alexandre Tkatchenko, and O. Anatole von Lilienfeld. Non-covalent interactions across organic and biological subsets of chemical space: Physics-based potentials parametrized from machine learning. *J. Chem. Phys.*, 148(24):241706, 2018.

[26] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98(14):146401(4), 2007.

[27] Jörg Behler. Neural network potential-energy surfaces for atomistic simulations. *Chem. Modell.*, 7(1):1–41, 2010.

[28] Kristoff T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus-Robert Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.*, 8:13890, 2017. arXiv:1609.08259.

[29] Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *NIPS 2017*, pages 991–1001, 2017.

[30] Justin S. Smith, Olexandr Isayev, and Adrian E. Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.*, 8:3192–3203, 2017.

[31] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML 2017*, 2017.

[32] Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M. Anderson, and Risi Kondor. Predicting molecular properties with covariant compositional networks. *J. Chem. Phys.*, 148(24):241745, 2018.

[33] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120:143001, 2018.

[34] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. arXiv:1802.08219, 2018.

[35] Eugen Merzbacher. *Quantum mechanics*. Wiley, 1998.

[36] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 389(20):457–484, January 1927.

[37] Walter Kohn. Two applications of the variational method to quantum mechanics. *Phys. Rev.*, 71:635–637, May 1947.

[38] V. R. Pandharipande and H. A. Bethe. Variational method for dense systems. *Phys. Rev. C*, 7:1312–1328, Apr 1973.

[39] P. Geerlings, F. De Proft, and W. Langenaeker. Conceptual density functional theory. *Chemical Reviews*, 103(5):1793–1874, 2003. PMID: 12744694.

[40] Kieron Burke. Perspective on density functional theory. *The Journal of Chemical Physics*, 136(15):150901, 2012.

[41] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. Schnet – a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[42] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[43] Michael Eickenberg, Georgios Exarchakis, Matthew Hirn, and Stéphane Mallat. Solid harmonic wavelet scattering: Predicting quantum molecular energy from invariant descriptors of 3D electronic densities. In *NIPS 2017*, pages 6540–6549, 2017.

[44] Michael Eickenberg, Georgios Exarchakis, Matthew Hirn, Stéphane Mallat, and Louis Thiry. Solid harmonic wavelet scattering for predictions of molecule properties. *J. Chem. Phys.*, 148:241732, 2018.

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[46] Stéphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition, 2008.

[47] Sigeru Huzinaga. Gaussian-type functions for polyatomic systems. i. *The Journal of Chemical Physics*, 42(4):1293–1302, 1965.

[48] Robert F. Stewart. Small gaussian expansions of slater-type orbitals. *The Journal of Chemical Physics*, 52(1):431–438, 1970.

[49] Stéphane Mallat. Group invariant scattering. *Comm. Pure Appl. Math.*, 65(10):1331–1398, October 2012.

[50] Thomas Wiatowski and Helmut Bölcskei. Deep convolutional neural networks based on semi-discrete frames. In *Proceedings of IEEE International Symposium on Information Theory*, pages 1212–1216, 2015.

[51] Philipp Grohs, Thomas Wiatowski, and Helmut Bölcskei. Deep convolutional neural networks on cartoon functions. In *IEEE International Symposium on Information Theory*, pages 1163–1167, 2016.

[52] Thomas Wiatowski and Helmut Bölcskei. A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Transactions on Information Theory*, 64(3):1845–1866, 2018.

[53] Wojciech Czaja and Weilin Li. Analysis of time-frequency scattering transforms. *Applied and Computational Harmonic Analysis*, 2017. In press.

[54] Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. In *Proceedings of the ISMIR 2011 conference*, pages 657–662, 2011.

[55] Joakim Andén and Stéphane Mallat. Scattering representation of modulated sounds. In *Proceedings of the DAFx 2012 conference*, 2012.

[56] Joan Bruna and Stéphane Mallat. Audio texture synthesis with scattering moments. arXiv:1311.0407, 2013.

[57] Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, August 2014.

[58] Vincent Lostanlen and Stéphane Mallat. Wavelet scattering on the pitch spiral. In *Proceedings of the 18th International Conference on Digital Audio Effects*, pages 429–432, 2015.

[59] Joan Bruna and Stéphane Mallat. Classification with scattering operators. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1561–1566, 2011.

[60] Laurent Sifre and Stéphane Mallat. Combined scattering for rotation invariant texture analysis. In *Proceedings of the ESANN 2012 conference*, 2012.

[61] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, June 2013.

[62] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, August 2013.

[63] X. Chen, X. Cheng, and S. Mallat. Unsupervised deep Haar scattering on graphs. In *Conference on Neural Information Processing Systems (NIPS)*, Montreal, Quebec, Canada, 2014.

[64] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for texture classification. arXiv:1403.1687, 2014.

[65] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *Proceedings in IEEE CVPR 2015 conference*, 2015. arXiv:1412.8659.

[66] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proceeding of the IEEE International Conference on Computer Vision*, pages 5619–5628, 2017.

[67] Matthew Hirn, Nicolas Poilvert, and Stéphane Mallat. Quantum energy regression using scattering transforms. arXiv:1502.02077, 2015.

[68] Matthew Hirn, Stéphane Mallat, and Nicolas Poilvert. Wavelet scattering regression of quantum chemical energies. *MMS*, 15(2):827–863, 2017.

[69] Thomas Blumensath and Mike E. Davies. On the difference between orthogonal matching pursuit and orthogonal least squares. Online at `http://eprints.soton.ac.uk/142469/1/BDOMPvsOLS07.pdf`, 2007.

[70] Thomas P Senftle, Sungwook Hong, Md Mahbubul Islam, Sudhir B Kylasa, Yuanxia Zheng, Yun Kyung Shin, Chad Junkermeier, Roman Engel-Herbert, Michael J Janik, Hasan Metin Aktulga, Toon Verstraelen, Ananth Grama, and Adri C T van Duin. The reaxff reactive force-field: development, applications and future directions. *npj Comput. Mater.*, 2(1):15011, 2016.

[71] Ying Yuan, Gregory Houchins, Pin-Wen Guan, and Venkatasubramanian Viswanathan. Uncertainty quantification of first principles computational phase diagram predictions of li-si system via bayesian sampling. arXiv:2003.13393 [cond-mat.mtrl-sci], 2020.

[72] Michael W. Swift and Yue Qi. First-principles prediction of potentials and space-charge layers in all-solid-state batteries. *Phys. Rev. Lett.*, 122:167701, Apr 2019.

[73] Song-Mao Liang, Franziska Taubert, Artem Kozlov, Jürgen Seidel, Florian Mertens, and Rainer Schmid-Fetzer. Thermodynamics of li-si and li-si-h phase diagrams applied to hydrogen absorption and li-ion batteries. *Intermetallics*, 81:32 – 46, 2017.

[74] Nongnuch Artrith, Alexander Urban, and Gerbrand Ceder. Constructing first-principles phase diagrams of amorphous Li_xSi using machine-learning-assisted sampling with an evolutionary algorithm. *J. Chem. Phys.*, 148:241711, 2018.

[75] T. D. Hatchard and J. R. Dahn. Study of the electrochemical performance of sputtered $Si_{1-x}Sn_x$ films. *J. Electrochem. Soc*, 151(10):A1628, 2004.

[76] Berk Onat, Ekin D. Cubuk, Brad D. Malone, and Efthimios Kaxiras. Implanted neural network potentials: Application to Li-Si alloys. *Phy. Rev. B*, 97:094106, 2018.

[77] Xavier Brumwell, Paul Sinz, Kwang Jin Kim, Yue Qi, and Matthew Hirn. Steerable wavelet scattering for 3D atomic systems with application to Li-Si energy prediction. In *NeurIPS Workshop on Machine Learning for Molecules and Materials, Montreal, Canada*, 2018.

[78] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *Phys. Rep.*, 810:1–124, 2019.

[79] H. Jónsson, G. Mills, and K. W. Jacobsen. Nudged elastic band method for finding minimum energy paths of transitions. In B. J. Berne, G. Ciccotti, and D. F. Coker, editors, *Classical and Quantum Dynamics in Condensed Phase Simulations*, page 385. World Scientific, 1998.

[80] D. Sholl and J.A. Steckel. *Density Functional Theory: A Practical Introduction*. Wiley, 2009.

[81] Xiao-Gang Lu, Malin Selleby, and Bo Sundman. Calculations of thermophysical properties of cubic carbides and nitrides using the debye–grüneisen model. *Acta Mater.*, 55(4):1215 – 1226, 2007.

[82] Pin-Wen Guan, Gregory Houchins, and Venkatasubramanian Viswanathan. Uncertainty quantification of DFT-predicted finite temperature thermodynamic properties within the Debye model. *J. Chem. Phys.*, 151(24):244702, 2019.

[83] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O. Anatole von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller. Assessment and validation of machine learning methods for predicting molecular atomization energies. *J. Chem. Theory Comput.*, 9(8):3404–3419, 2013.

[84] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8(1), 2017.

[85] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[86] Stefan Chmiela, Huziel E. Sauceda, Igor Poltavsky, Klaus-Robert Müller, and Alexandre Tkatchenko. sgdml: Constructing accurate and data efficient molecular force fields using machine learning. *Computer Physics Communications*, 240:38–45, 2019.

[87] Anders S. Christensen, Lars A. Bratholm, Felix A. Faber, and O. Anatole Von Lilienfeld. Fchl revisited: Faster and more accurate quantum machine learning. *The Journal of Chemical Physics*, 152(4):044107, 2020.

[88] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *CoRR*, abs/2102.03150, 2021.

[89] David Peter Kovacs, Cas van der Oord, Jiri Kucera, Alice Allen, Daniel Cole, Christoph Ortner, and Gabor Csanyi. Linear atomic cluster expansion force fields for organic molecules: beyond rmse. *ChemRxiv*, 2021.

[90] Andrew R. Leach. *Molecular modeling: principles and applications*. Prentice Hall, 2001.

[91] V Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187–207, 1985.

[92] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling, 2019.

[93] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[94] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[95] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3538–3545. AAAI Press, 2018.

[96] Ladislav Rampášek and Guy Wolf. Hierarchical graph neural nets can capture long-range interactions, 2021.