A COMPUTATIONAL INVESTIGATION OF SOLID ELECTROLYTE MATERIALS USING FIRST PRINCIPLES AND MACHINE LEARNING METHODOLOGIES

BY

D. CORY LYNCH

A Dissertation submitted to the Graduate Faculty of

WAKE FOREST UNIVERSITY GRADUATE SCHOOL OF ARTS AND SCIENCES

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Physics

August 2026

Winston-Salem, North Carolina

Approved By:

Natalie Holzwarth, Ph.D., Advisor

Fan Yang, Ph.D., Chair

Samuel Cho, Ph.D.

Timo Thonhauser, Ph.D.

Stephen Winter, Ph.D.

Acknowledgments

First and foremost, I'd like to thank my advisor, Dr. Natalie Holzwarth, for her continued support and collaboration during the course of my graduate studies here at Wake Forest, including being accommodating with regards to personal health issues I have faced throughout. Thanks to her impeccable attention to detail and inquisitive nature during research discussions, I have been able to grow as a researcher and physicist. I would like to thank my committee members, Dr. Fan Yang, Dr. Samuel Cho, Dr. Timo Thonhauser, and Dr. Stephen Winter, for taking the time out of their busy schedules to read the contents of this work, give their helpful suggestions and comments, and support me during the final stretch of the dissertation process.

I would also like to think several other people who have been indispensable during my studies at Wake Forest. Dr. Yan Li, a recent graduate and excellent research coauthor, helped me greatly during my initial familiarization with several of the software packages we use when I first joined our research group, and has been very supportive throughout. Chuin Wei Tan, a research coauthor at Harvard University, has been incredibly helpful troubleshooting the Allegro software and providing machine learning related assistance and advice during our current project. The DEAC support team, and in particular, Dr. Sean Anderson, was essential for his assistance in getting software packages installed and troubleshooting many technical issues that came up during our research.

I'd also like to thank several of my professors at the University of North Carolina Greensboro, including Dr. William Gerace and Dr. Ian Beatty, both of whom introduced me to the joy of physics. Thanks to their unique teaching philosophies, their courses continuously challenged me to become a better thinker and problem solver, pushing me to never stop asking questions and looking for new insights into the mechanisms of the physical world around us. Additionally, Dr. Ratnasingham Shivaji and Dr. Dan Yasaki were integral to my mathematical studies, showing me that abstraction itself is quite beautiful, and that mathematics is much larger and more interconnected than I had previously been able to appreciate.

Finally, I would like to thank my parents, Kenneth and Susan, my siblings, Angie, Kenneth, and Renee, and my fiancee, Crystal. They have all have been incredibly supportive of my studies and encouraged me to continue working toward my goals even when things became challenging. They have helped me deal with many obstacles I have faced personally throughout my educational career and been by my side no matter what. Without them, I would not be where I am today.

Table of Contents

Acknowledgments ii

List of Figures vi

List of Tables xi

List of Abbreviations xii

Abstract xiii

Introduction xv

Chapter 1 Background and Motivation 1
- 1.1 Background on Lithium Ion Batteries 2
  - 1.1.1 Conventional Lithium Ion Batteries 3
  - 1.1.2 Solid State Lithium Ion Batteries 4
- 1.2 Boracite and Thioboracite Solid Electrolytes 5
  - 1.2.1 Lithium Boracite – $Li_4B_7O_{12}Cl$ 6
  - 1.2.2 Partial Aluminum Substitution – $Li_4Al_3B_4O_{12}Cl$ 8
  - 1.2.3 Lithium Thioboracite – $Li_4B_7S_{13}I$ 10
- 1.3 Lithium Phosphate Solid Electrolytes 12
  - 1.3.1 Lithium Phosphate/Silicate Alloy - $Li_{3.25}P_{0.75}Si_{0.25}O_4$ 13
- 1.4 Lithium Argyrodite Solid Electrolytes 14
  - 1.4.1 Lithium Argyrodite - Li_6PS_5Cl 15
  - 1.4.2 ZnO-doped Lithium Argyrodite - $Li_{5.5}PS_{4.75}Cl\,[ZnO]_{0.25}$ 16

Chapter 2 Select First Principles Methods 18
- 2.1 Quantum Theory of Crystals 18
  - 2.1.1 Born Oppenheimer Approximation 19
  - 2.1.2 Density Functional Theory 24
  - 2.1.3 Hellmann Feynmann Theory and Interatomic Forces 32
  - 2.1.4 Periodic Lattices and Bloch Theory 34
  - 2.1.5 Pseudopotentials 39
- 2.2 Phonons in Crystals 40
  - 2.2.1 Finite Differences Method 41

2.3 Transport Properties of Crystals 43
2.3.1 Molecular Dynamics Simulations 43
2.3.2 Calculating Ionic Conductivity 47

Chapter 3 Select Machine Learning Methods 50
3.1 Introduction to Neural Networks 50
3.1.1 Feed Forward Neural Networks 51
3.1.2 Activation Functions 54
3.1.3 Loss Functions 56
3.1.4 Optimization and Backpropagation 57
3.1.5 Input Features 62
3.1.6 Neural Network Flexibility and Limitations 63
3.2 Graph Neural Networks 65
3.2.1 Graphs 65
3.2.2 Feature Embedding 67
3.2.3 Message Passing and Pooling Layers 67
3.2.4 Message Passing Graph Neural Network 70
3.3 Simple Examples 71
3.3.1 Feed Forward Neural Network 71
3.3.2 Message Passing Graph Neural Network 73

Chapter 4 Neural Networks in Physics 79
4.1 Group Theory 79
4.1.1 Definition of a Group 80
4.1.2 Group Representation Theory 80
4.1.3 Invariance and Equivariance 82
4.1.4 Continuous Groups and E(3) 83
4.1.5 Tensor Products 84
4.2 Selected Example Neural Network Architectures 88
4.2.1 Behler and Parrinello Neural Network 88
4.2.2 MEGNET Graph Network Architecture 90
4.3 Adopted Neural Network Architectures 93
4.3.1 NequIP Architecture 93
4.3.2 Allegro Architecture 97

Chapter 5 Research Project Summaries 102
5.1 Computational investigation of the structural and electrolyte properties of the extended family of lithium (thio)boracite materials: $Li_4B_7O_{12}Cl$ and beyond 102

5.1.1 Optimized Static Lattice $R3c$ Structures 102
5.1.2 Optimized Static Lattice Cc Structures 104
5.1.3 Optimized Static Lattice $F\bar{4}3c$ Structures 107
5.1.4 Phonon Calculations 109
5.2 A study of the use of machine learning methodology for computing ionic conductivity in an example solid-state electrolyte material . . . 112
5.2.1 Test System and Initial Allegro Familiarization 113
5.2.2 Molecular Dynamics Stability Issues 117
5.2.3 Allegro Model Analysis 119
5.2.4 Assessment of Structural Prediction Capability 120
5.2.5 Molecular Dynamics and Conductivity Estimation 122
5.2.6 Comparison of Single vs. Combined Structural Training 126
5.3 Estimating the ionic conductivity of select (thio)boracite solid electrolyte materials using the Allegro machine learned interatomic potential to extend first principles calculations 128
5.3.1 Project Scope and Objectives 129
5.3.2 Computational Details and Timeline 130
5.4 A Study of Silicon Doped Argyrodite in Collaboration with Argonne National Laboratory 130
5.4.1 Determining the Optimal Substitution Sites for ZnO 130
5.4.2 Computed X-Ray Diffraction Patterns 133

Appendix A Selected Practical Computational Details and Example Code 135
A.1 Pytorch Neural Network Example 135
A.2 Close-Pair Dataset Construction 144

Appendix B Publication: Computational investigation of the structural and electrolyte properties of the extended family of lithium (thio)boracite materials: $Li_4B_7O_{12}Cl$ and beyond 153

Appendix C Preprint: A study of the use of machine learning methodology for computing ionic conductivity in an example solid-state electrolyte material 175

Bibliography 206

Curriculum Vitae 219

List of Figures

1.1 The left figure depicts demand for lithium batteries from 2020 to 2025 with demand projection for 2026, which is based off of data in the BloombergNEF Electric Vehicle Outlook 2026 report [1]. The right figure depicts the number of peer-reviewed publications from 2016 to 2025 relating to solid-state batteries, which was obtained through a Web of Science search for topic "solid-state batter*". 2

1.2 Figure depicting a conventional lithium ion battery with a liquid electrolyte and separator, as reproduced from Zhang et al. [10] under the Creative Commons 3.0 license. 3

1.3 Figure comparing a conventional lithium ion battery to a theoretical solid-state lithium ion battery, as reproduced from Joshi et al. [13] under Creative Commons License. 5

1.4 Figure showing the conventional cell of the disordered cubic $F\bar{4}3c$ structure of $Li_4B_7O_{12}Cl$ as reproduced from reference [15] with permission. In this figure Li, B, O, and Cl are represented by blue, black, red, and green spheres. 7

1.5 Figure showing Arrhenius plots for $Li_4B_7O_{12}Cl$ and $Li_5B_7O_{12.5}Cl$ as reproduced from reference [15] with permission. The $Li_4B_7O_{12}Cl$ computational results are in blue and are compared to various experimental results in cyan, black, and orange. The $Li_5B_7O_{12.5}Cl$ computational results are in red and are compared to experimental results in green. . 8

1.6 Figure showing the conventional cell of the disordered cubic $F\bar{4}3c$ structure of $Li_4Al_3B_4O_{12}Cl$ as reproduced from reference [19] with permission. Here the red, large green, and small green spheres denote O, Cl, and Li, and the green triangles and grey tetrahedra denote BO_3 and MO_4 units. 9

1.7 Figure showing Arrhenius plots of the ionic conductivity of $Li_4Al_3B_4O_{12}Cl$ as reproduced from reference [19] with permission. Here the different colored datapoints represent different doping ratios based on the choice of y during the synthesis procedure. 10

1.8 Figure showing the cell of the cubic $F\bar{4}3c$ and tetragonal $I\bar{4}c2$ structures of $Li_6B_7S_{13}I$ as reproduced from reference [20] with permission. Here the purple and green spheres represent I and Li, and the blue tetrahedra represent BS_4 units. 11

1.9 Figure showing Arrhenius plots of the ionic conductivity of $Li_6B_7S_{13}I$ as reproduced from reference [20] with permission. The figure on the left shows the data from first principles simulations and the figure on the right shows data from EIS measurements. 12
1.10 Figure showing the structure of Li_3PO_4 + Li_4SiO_4 based on the cif file as published by Deng et al. [23]. Here the green, blue, red, and grey coloring of the spheres represent lithium, silicon, phosphorous, and oxygen. The lithium sites are fractionally occupied and the phosphorous/silicon sites are occupied at a ratio of 3:1. 13
1.11 Figure showing MSD plots of the lithium ions of various alloys of Li_3PO_4 + Li_4SiO_4 as reproduced from reference [23] with permission. 14
1.12 Figure showing a section of the crystalline structure of Li_6PS_5I as reproduced from reference [24] with permission. 15
1.13 Figure showing an Arrhenius plot of Li_6PS_5Cl derived from impedence spectroscopy at 309K as reproduced from reference [25] with permission. 16

2.1 Figure depicting a Bravais lattice with primitive lattice vectors $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$. 35
2.2 Figure depicting the primitive (left) and conventional cell (right) of $NaCl$ in the face centered cubic crystalline form described by the $Fm\bar{3}m$ space group. The yellow and green spheres represent sodium and chlorine. The cif files used to prepare these figures were obtained through the materials project database [42]. 36

3.1 Plots of the logistic (left) and hyperbolic tangent (right) activation functions. 55
3.2 Plots of the ReLU (left) and SiLU (right) activation functions. 55
3.3 Example of a visual graph structure and its mathematical set representation. 66
3.4 A visual example of graph pooling operations, where the arrows represent pooling layers. After the first pooling layer, we see that a colored subgraph of the original graph G is now being represented by a single vertex of the same color in the new output graph G'. The second pooling layer is a global pooling layer, whose output graph G'' is now simply a single vertex that represents the input graph in its entirety. . 70
3.5 Simple FFNN training example demonstrating the flexibility of the neural network construction for learning complicated real valued functional forms from training datasets. Here the data represents the distance between two like charges and their potential energy. 74
3.6 Computational flowchart of the simple architecture we have defined to demonstrate the functionality of MP-GNNs. 77

4.1 Visualization of the FFNN architecture as reproduced from reference [77] with permission. 89
4.2 Computed RDF functions for various methods (left) and energy per atom errors of the NN approach compared to DFT (right), as reproduced from reference [77] with permission. 91
4.3 Visualization of the NequIP architecture as reproduced from reference [78] with permission. 94
4.4 Visualization of the Allegro architecture as reproduced from reference [79] with permission. 97
4.5 Comparison of Li_3PO_4 MSD results between an Allegro potential model and FPMD, as reproduced from reference [79] with permission. 100

5.1 $R3c$ distorted conventional cell structures for the $Li_4B_7O_{12}Cl$ (left) and $Li_4Al_3B_4O_{12}Cl$ (right). 103
5.2 $R3c$ distorted conventional cell structures for the $Li_4B_7S_{12}Cl$ (left) and $Li_4Al_3B_4S_{12}Cl$ (right). 103
5.3 $R3c$ distorted conventional cell structures for the $Li_6B_7O_{13}Cl$ (left) and $Li_6Al_3B_4O_{13}Cl$ (right). 105
5.4 $R3c$ distorted conventional cell structures for the $Li_6B_7S_{13}Cl$ (left) and $Li_6Al_3B_4S_{13}Cl$ (right). 105
5.5 Conventional cell structures for $Li_6B_7S_{13}Cl$ in $R3c$ (left) and Cc (right) configurations. 106
5.6 Cc conventional cell structures for the $Li_6Al_3B_4O_{13}Cl$ and $Li_6Al_3B_4S_{13}Cl$ configurations. 106
5.7 Conventional cell structures for $Li_4B_7S_{12}Cl$ in $R3c$ (left) and ordered $F\bar{4}3c$ (right) configurations. 108
5.8 Band structure and PDOS plots for the $Li_4B_7O_{12}Cl$ $R3c$ configuration (left) and the $Li_4B_7S_{12}Cl$ $F\bar{4}3c$ configuration (right). 110
5.9 Band structure and PDOS plots for the $Li_4Al_3B_4O_{12}Cl$ $R3c$ configuration (left) and the $Li_4Al_3B_4S_{12}Cl$ $R3c$ configuration (right). 111
5.10 Band structure and PDOS plots for the $Li_6B_7O_{13}Cl$ $R3c$ configuration (left) and the $Li_6B_7S_{13}Cl$ Cc configuration (right). 111
5.11 Band structure and PDOS plots for the $Li_6Al_3B_4O_{13}Cl$ Cc configuration (left) and the $Li_6Al_3B_4S_{13}Cl$ Cc configuration (right). 112
5.12 Figure depicting the relaxed electronic ground-state structures for two configurations of $Li_{3.25}Si_{0.25}P_{0.75}O_4$: structure S1 (left) and structure S2 (right). Green, blue, red, and white spheres represent Li, Si, P, and O atoms respectively. 113

5.13 Figure depicting the impact of the choice of r_{cut} on the model's training E_{MAE} error for a fixed model "size" (choice of hyperparameters) and input dataset size (number of configurations). Here r_{cut} is given in $\mathring{A}$ and E_{MAE} is given in eV. E_{MAE} is defined in equation 53 of the preprint manuscript C. 114
5.14 Dataset size test with 2500, 5000, and 7500 input dataset sizes, showing average training E_{MAE} (left) and F_{MAE} (right) error vs. input dataset size (number of configurations), with error defined by the standard deviation. Here E_{MAE} is given in eV and F_{MAE} is given in eV/$\mathring{A}$. F_{MAE} is defined in equation 54 of the preprint manuscript C. 116
5.15 Hyperparameter test with "small", "medium", and "large" model sizes, showing average training (left) and validation (right) weighted sum values vs. model size, with error defined by the standard deviation. The weighted sum values are unitless, and are defined in equation 56 of the preprint manuscript C. 116
5.16 Figure showing example temperature plots, depicting model instability during a molecular dynamics run (left) compared to a stable molecular dynamics run (right), both at target temperatures of 900K. 119
5.17 Structure S1 (red) and S2 (blue) volume change (left) and energy change (right) vs Allegro model validation weighted sum error. Volume changes are given in $\mathring{A}^3$, energy changes are given in meV, and weighted sum values are unitless. In these plots, the data is given by the unfilled markers and the linear fit to the data is given by the lines. 121
5.18 Figure showing the interval functions $\overline{\Lambda}^{\text{full}}_{\text{CS2},v}(\tau)$ with error shading (top left), $\overline{\overline{\Lambda}}^{\text{full}}_{\text{CS2}}(\tau)$ with error shading (top right), $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ without error shading (bottom left), and $\overline{\overline{\overline{\Lambda}}}^{\text{full}}(\tau)$ with error shading (bottom right) for structure S1. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve. 123
5.19 Figure showing the interval functions $\overline{\Lambda}^{\text{tr}}_{\text{CS2},v}(\tau)$ with error shading (top left), $\overline{\overline{\Lambda}}^{\text{tr}}_{\text{CS2}}(\tau)$ with error shading (top right), $\overline{\overline{\Lambda}}^{\text{tr}}_{\Phi}(\tau)$ without error shading (bottom left), and $\overline{\overline{\overline{\Lambda}}}^{\text{tr}}(\tau)$ with error shading (bottom right) for structure S1. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve. 124
5.20 Arrhenius plots of full (left) and tracer (right) conductivity results for each of the Allegro models CS2 - CS6 for structure S1. 125
5.21 Arrhenius plots of full (left) and tracer (right) conductivity results for each of the Allegro models CS2 - CS6 for structure S2. 125

5.22 Figure showing the interval functions $\overline{\Lambda}^{\text{full}}_{\text{SS1},v}(\tau)$ with error shading (left), and $\overline{\overline{\Lambda}}^{\text{full}}_{\text{SS1}}(\tau)$ with error shading (right) for structure S1. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve. 127

5.23 Figure showing the interval functions $\overline{\Lambda}^{\text{full}}_{\text{SS2},v}(\tau)$ with error shading (left), and $\overline{\overline{\Lambda}}^{\text{full}}_{\text{SS2}}(\tau)$ with error shading (right) for structure S2. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve. 127

5.24 Allegro model SS1 (left) and SS2 (right) average Arrhenius plots of full and tracer conductivity results for structures S1 and S2 respectively. Combined CS2 - CS6 model average results are included for comparison. Tracer conductivity estimates are given in red and full conductivity estimates are given in blue. 128

5.25 Idealized candidate cubic structures for $Li_{24}P_4S_{20}Cl_4$ (left), $Li_{22}P_4S_{19}Cl_4ZnO$ (center), and $Li_{21}P_4S_{18}Cl_5ZnO$ (right). Here blue, magenta, grey, yellow, red, and green spheres represent lithium, phosphorous, silicon, sulfur, oxygen, and chlorine. 133

5.26 X-ray results for all three idealized cubic structures determined computationally (left) and the idealized $Li_{22}P_4S_{19}Cl_4ZnO$ cubic structure's computed x-ray data compared to its experimental counterpart (right). 133

List of Tables

2.1 Table depicting the Wyckoff positions of the primitive and conventional cells of $NaCl$ described by the $Fm\bar{3}m$ space group, based on information from the materials project database [42]. 37

5.1 Table defining Allegro model "sizes", which determine the specific hyperparameter choices for the model. 115

5.2 List of 10 different Allegro models studied in this work. Note that the labels serve as a compact notation for the model definition, in the format XYZ, where X is the input dataset label (C for "combined", S for "single"), Y is the model size label (S for "small", M for "medium" as defined in table 5.1), and Z is simple a numerical label for further model differentiation, such as different model initialization. 120

5.3 Computational details for the current project. 130

5.4 Table detailing the properties of the three candidate structures for the lithium argyrodite materials investigated in this study, $Li_{24}P_4S_{20}Cl_4$, $Li_{22}P_4S_{19}Cl_4ZnO$, and $Li_{21}P_4S_{18}Cl_5ZnO$. Since the structures are cubic, there is only one lattice parameter to define, which is a, given in Angstroms. The precursor materials which serve as potential decomposition products are listed, and the formation energy is given with respect to these materials, indicating stability with respect to decomposition. 132

List of Abbreviations

Abbreviation	**Meaning**
DFT	Density Functional Theory
MLIP	Machine Learned Interatomic Potential
QE	QUANTUM ESPRESSO
FPMD	First Principles Molecular Dynamics
GNN	Graph Neural Network
MLP	Multi Layer Perceptron
FFNN	Feed Forward Neural Network

Abstract

In the midst of a rapid transition away from fossil fuels to cleaner technologies including renewable energy and electrified transportation, lithium battery demand is surging. Global demand for lithium-ion batteries is expected to reach 2 TWh/year in 2026, which corresponds to a doubling of demand in just 3 short years. Over 75% of this demand is attributed to electric vehicle production, and nearly the rest of the demand comes from stationary storage applications. Electric transportation requires batteries with high energy densities which can charge quickly, with excellent safety profiles and long lifespans. While current lithium-ion batteries excel in many applications including stationary storage and consumer electronics, implementations in electric vehicles expose some of their limitations.

Current generation lithium-ion batteries are composed of two electrodes, the anode and the cathode, and a liquid or gel electrolyte. The electrolyte serves as a charge carrier for mobile lithium-ions across the cell. Typical electrolytes contain toxic and flammable materials and have thermally limited charge and discharge rates. Furthermore, the relatively large volume required of liquids and gels limit the energy density and capacity of the batteries. Solid-state batteries are a next-generation technology where the liquid or gel electrolyte is replaced by a solid electrolyte, which theoretically would allow for higher energy densities, capacities, and charging/discharging rates. Research in this area is expanding quickly, and the study of solid electrolytes is crucial for the development of solid-state batteries.

In this work, we first began investigating a family of eight boracite and thioboracite solid electrolyte materials. Our research discovered many interesting properties such as energetically favorable structural configurations, dynamic stability, and chemical stability with respect to decompositions as well as in contact with pure lithium metal. We then attempted to study the ionic conductivity of these materials using the typical method involving first principles molecular dynamics simulations.

Ionic conductivity is a macroscopic property which measures how easily ions can flow through a material. We can estimate this property from microscopic phenomena by determining the average ionic diffusion through the crystal over time. To do so accurately, we must average over a large ensemble of systems, which in practice is done through the averaging of samples from long molecular dynamics simulations. Unfortunately, computational limitations prevented us from running long enough simulations to quantitatively assess the ionic conductivity. As a result, we shifted our attention to machine learning approaches which extend first principles methods to overcome these limitations.

Our next project involved using a test system comprised of two different configurations of a lithium phosphate / lithium orthosilicate alloy to assess the capabilities of the Allegro machine learned interatomic potential architecture for this purpose. In this study, we were able to train several Allegro models which all had good agreement with the DFT generated training and validation data as evidenced by weighted sum metrics on the order of 10^{-3}.

Long time molecular dynamics simulations were found to be quite stable overall, though infrequent instabilities in the simulations brought to our attention known issues with model interpolation at close atomic pair distances. Our attempts to mitigate this issue through the manual addition of training configurations with closer atomic pair qualitatively seemed to lower the number of failures, as indicated by a simple 900K screening test which checks for such occurrences.

We were also able to show that full conductivity estimates at low temperature were well controlled in general when calculated from long 2.8 ns simulations using trained Allegro models, though high temperatures were not. We also found that while the choice of the initial velocity distribution greatly impacts the resultant conductivity estimation, the block averaging and Allegro model variance were not an issue.

Thanks to the successful outcomes from this study using Allegro with the simpler test system, we are now in the early stages of a new project where we are planning to continue the investigation of the boracite materials to quantitatively determine their ionic conductivity. This project is ongoing and will hopefully be a future publication, and therefore it will not be included here.

Introduction

This introduction will serve to outline the contents of the dissertation.

- Chapter 1 covers the background information which has informed my decision to focus on solid-state electrolytes as the material category of study in my work, and presents brief motivating background information on specific materials I have chosen to study
- Chapter 2 covers the main theoretical framework of quantum mechanics which are used to computationally investigate these interesting materials.
- Chapter 3 introduces the main theoretical framework of machine learning, specifically neural networks, which are used to extend so-called "first principles" calculations.
- Chapter 4 details how this extension is performed, integrating the more general neural network construction with quantum mechanics theory to expand the computational tools for investigating the materials I wish to learn more about.
- Chapter 5 summarizes the various research projects I have undertaken to explore these materials of interest, focusing on my contributions to the projects and what has been learned from these investigations.
- Appendix: The Appendix contains attached published works and preprint manuscripts which resulted from the research discussed in Chapter 5, as well as some example Python code which was authored by me in the course of the research.

Chapter 1: Background and Motivation

In this chapter, I briefly describe the current state of lithium battery technology and research, inspiring my decision to focus my efforts on investigating solid-state electrolytes. I then introduce previous works which have studied several interesting and promising boracite and thioboracite materials and motivated me to focus a large part of my research on studying a small family of eight known and predicted boracite and thiobracite materials. Work involving the investigation of ground-state structures, phonon spectra, and stability of these materials in collaboration with former graduate student Dr. Yan Li and my advisor Dr. Natalie Holzwarth are presented in section 5.1. Our current project focusing on the ionic conductivity of a select subset of these materials using the machine learned interatomic potentials, though still in early stages, is outlined in section 5.3.

I then discuss two solid electrolyte materials which were only briefly investigated. The first material is an alloy of lithium phosphate and lithium silicate which was used only as a test material for a methodological investigation involving machine learning techniques, and not for material investigation, in collaboration with Chuin Wei Tan, a Harvard Graduate Student. The results from this collaboration are presented in section 5.2. The second material is a ZnO substituted lithium argyrodite which was studied in collaboration with experimental work performed by Zachary D. Hood, a researcher at Argonne National Laboratory, and Udochukwu "Udo" Eze, a Georgia Tech graduate student. Unfortunately due to circumstances beyond my control, this work remains unpublished. The unpublished summary of my contributions to this project is presented in section 5.4.

1.1 Background on Lithium Ion Batteries

Lithium ion batteries are a type of rechargeable energy storage device with high energy density and good longevity. Global manufacturing of lithium ion batteries continues to climb exponentially due to increased demand for use in applications such as stationary energy storage for renewable power generation, electric vehicles, and consumer electronics [1] [2]. Many applications have high energy density and performance needs that are limited by the current lithium ion battery technology, which also has environmental issues surrounding manufacturing and disposal, public health issues involving safety, and market issues such as cost. Solid state lithium ion batteries are a proposed next generation technology which offers a solution to many of these limitations, and have experienced a dramatic increase in research interest over the last decade [3].

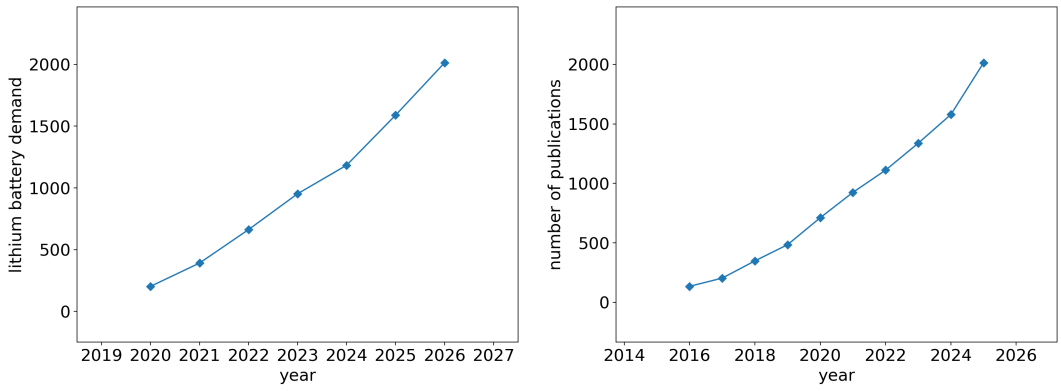


Figure 1.1: The left figure depicts demand for lithium batteries from 2020 to 2025 with demand projection for 2026, which is based off of data in the BloombergNEF Electric Vehicle Outlook 2026 report [1]. The right figure depicts the number of peer-reviewed publications from 2016 to 2025 relating to solid-state batteries, which was obtained through a Web of Science search for topic "solid-state batter*".

1.1.1 Conventional Lithium Ion Batteries

Groundbreaking lithium ion battery research by Goodenough, Whittingham, and Yoshino – for which they were awarded the Nobel Prize in Chemistry in 2019 – paved the way for the first commercial lithium ion batteries produced by Sony in 1991 [4]. Modern lithium batteries are further improvements upon these works, and typically consist of lithium cobalt oxide ($LiCoO_2$) cathodes and graphite (C_6) anodes. These electrodes are layered with a porous separator between them that is immersed in a non-aqueous liquid electrolyte which is a mixture of primarily lithium hexafluorophosphate ($LiPF_6$) as well as several other additives [5]. Although an incredible amount of research has been done to improve safety, performance, and longevity of these lithium batteries, such as the more recent lithium polymer battery development [6], all lithium batteries with liquid electrolytes are subject to safety issues, including poor recyclability of materials, toxic waste products, and safety issues regarding flammability and thermal runaway [7, 8, 9, 10].

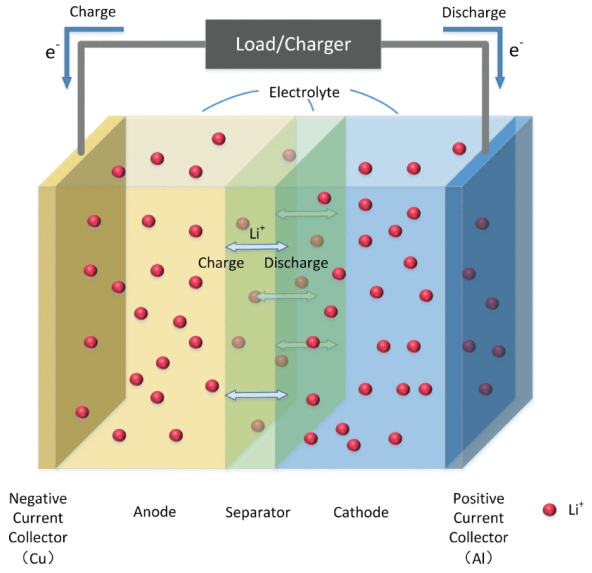


Figure 1.2: Figure depicting a conventional lithium ion battery with a liquid electrolyte and separator, as reproduced from Zhang et al. [10] under the Creative Commons 3.0 license.

1.1.2 Solid State Lithium Ion Batteries

Solid state lithium ion batteries, as the name would imply, are lithium ion batteries with solid electrolyte materials as opposed to the liquid electrolytes currently in use in lithium ion batteries today. Lacking a separator and using a solid electrolyte to directly connect to the electrodes allows the solid-state battery to offer higher energy densities and better performance. The lack of flammable and toxic liquid chemicals greatly improves safety and reduces toxic waste to make for a more environmentally friendly battery. Solid state batteries additionally have the potential to be recharged faster than current lithium ion batteries, and due to their increased energies densities allowing for smaller size, seem particularly suited for electric vehicle applications [11].

An ideal solid-state battery would have pure lithium metal as the anode, though alternative anode materials such as lithium-silicon metallic alloys have been proposed. The most common cathode materials include lithium metal oxides such as lithium cobalt oxide ($LiCoO_2$) and lithium manganese oxide ($LiMn_2O_4$). While the electrodes are certainly important, perhaps the most important component is the solid electrolyte material of choice. The electrolyte must have good lithium ion conductivity and negligible electrical conductivity. It must be stable in contact with the electrodes and have minimal dendrite formation, as well as flexibility to ensure good contact with the electrodes. Solid electrolytes being investigated are typically polymers, such as polyethylene oxide (C_2H_4O), or inorganic materials, such as lithium phosphorus oxynitride ($LiPON$). Other unconventional materials such as halides and chlorides may also be of interest [12].

One of the current issues involving the development of solid-state batteries is the fact that pure lithium metal has a serious issue of dendrite formation during cycling, leading to short circuits and capacity degradation. Another issue is loss of structural integrity and electrode contact due to volume expansion and contraction

during cycling. While there are many significant hurdles to overcome in order to realize a solid-state battery that can outperform the current conventional lithium ion batteries, the field is quite active in researching and developing new materials to overcome these issues [13].

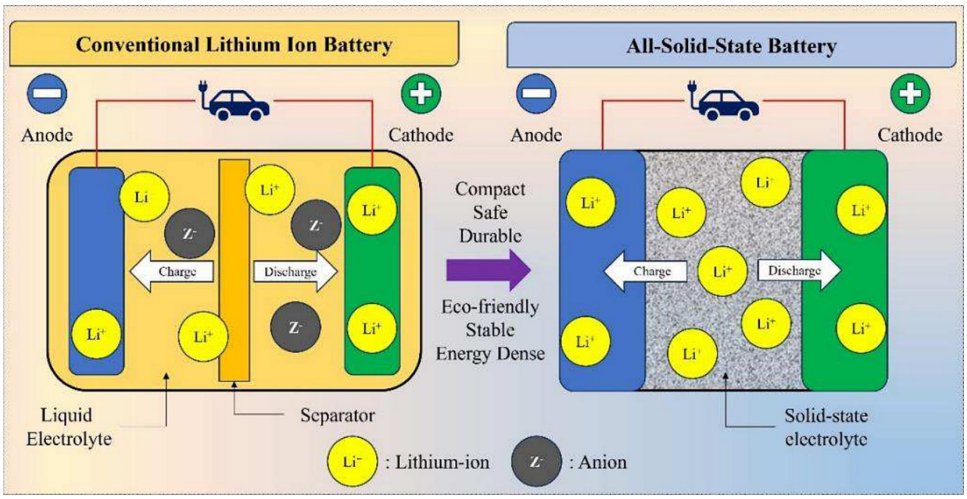


Figure 1.3: Figure comparing a conventional lithium ion battery to a theoretical solid-state lithium ion battery, as reproduced from Joshi et al. [13] under Creative Commons License.

The central component of the solid-state battery is the solid electrolyte material, which must interface with a pure lithum anode with good contact and stability, provide good lithium ion conduction across its volume, and also interface well with the cathode. As a result, in this work I focus my attention on the investigation of solid-state electrolytes only, and forego investigations into cathode or anode materials.

1.2 Boracite and Thioboracite Solid Electrolytes

Building upon the previous works published by our group [14] [15], I am largely investigating materials which are part of a family of lithium boracite and thioboracite chemical and structural composition. These materials are not only of interest for commercial application, but are of scientific interest due to their unique structural

geometry, ionic transport mechanisms, and understanding of their basic properties as well. Here I introduce some of the historical findings that have motivated the choice to continue studying these boracite and thioboracite materials.

1.2.1 Lithium Boracite – $Li_4B_7O_{12}Cl$

Several papers have been published experimentally and computationally investigating the lithium boracite $Li_4B_7O_{12}Cl$ as a promising solid-state electrolyte. In the early work in 1977 by Jeitschko, et al. [16], $Li_4B_7O_{12}Cl$ was synthesized and experimentally analyzed. Three distinct crystalline phases were discovered through the use of high temperature single crystal analysis: the α phase below 310K, the β phase between 310K and 348K, and the γ phase at temperatures above 348K. The α phase can be modelled as an ideal $F\overline{4}3c$ face centered cubic structure, with fractional occupancy of the lithium atoms at 32e Wyckoff positions. The conductivity was experimentally determined by analyzing a DC electrochemical cell consisting of the electrolyte sandwiched between two sections of lithium amalgum. The conductivity at room temperature was determined to be about 2×10^{-6} $S\ cm^{-1}$.

A more recent paper in 2019 by Tan et al. [17] has shown that by utilizing an ionothermal synthesis route, they were able to successfully synthesize and analyze a single-phase $Li_4B_7O_{12}Cl$ polycrystalline sample. Due to the high purity of the sample, they were able to get highly accurate electrochemical performance measurements for the disordered cubic phase of the material. They isostatically pressed powdered $Li_4B_7O_{12}Cl$ into a dense pellet and then attached two stainless steel disks on either side of the pellet in order to apply voltage across the sample. From this experimental setup, they were able to measure the ionic conductivity to be quite high at 3×10^{-4} $S\ cm^{-1}$ at room temperature, two orders of magnitude higher than earlier estimates. An even more recent paper in 2022 by Katsumata et al. [18] reported somewhat lower

conductivities of around 10^{-5} $S\ cm^{-1}$ at 294K.

Research in our group authored by Li and Holzwarth [15] computationally investigated the $Li_4B_7O_{12}Cl$ boracite material, determining the ground-state structure to have an ordered rhombohedral $R3c$ symmetry configuration. They were able to determine phonon band structures as well as chemical and dynamic stability of the material. From first principles molecular dynamics, they were able to estimate the conductivity to be about 10^{-4} $S\ cm^{-1}$ at 300K, in good agreement with experimental studies. Reproduced from their paper, we see a model of the conventional cell of the cubic $F\bar{4}3c$ disordered structure shown in Figure 1.4 and an Arrhenius plot shown in Figure 1.5. Notably, from the Arrhenius plot, it can be seen that the $Li_4B_7O_{12}Cl$ material has much higher lithium ionic conductivity than the $Li_5B_7O_{12.5}Cl$ material. We also see that the computational estimates are typically overestimating the conductivity determined by experimental studies at lower temperatures.

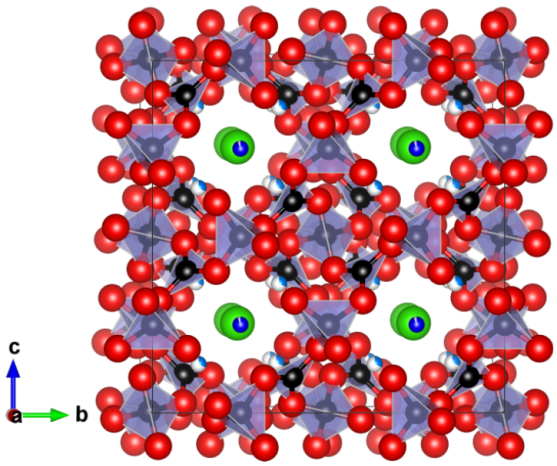


Figure 1.4: Figure showing the conventional cell of the disordered cubic $F\bar{4}3c$ structure of $Li_4B_7O_{12}Cl$ as reproduced from reference [15] with permission. In this figure Li, B, O, and Cl are represented by blue, black, red, and green spheres.

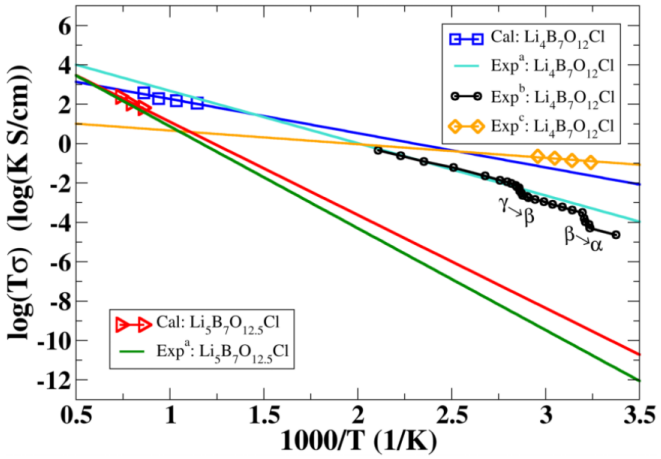


Figure 1.5: Figure showing Arrhenius plots for $Li_4B_7O_{12}Cl$ and $Li_5B_7O_{12.5}Cl$ as reproduced from reference [15] with permission. The $Li_4B_7O_{12}Cl$ computational results are in blue and are compared to various experimental results in cyan, black, and orange. The $Li_5B_7O_{12.5}Cl$ computational results are in red and are compared to experimental results in green.

1.2.2 Partial Aluminum Substitution – $Li_4Al_3B_4O_{12}Cl$

A recent paper by Kajihara et al. in 2017 [19] studied the impacts of partial replacement of boron sites in the lattice with either aluminum or gallium. This partial replacement was accomplished by replacing all of the tetrahedral BO_4 units from the framework of $Li_4B_7O_{12}Cl$ with either AlO_4 or GaO_4, creating a formula unit of $Li_4M_3B_4O_{12}Cl$, where $M = Ga, Al$. This replacement was achieved by preparing samples with the melt-quenching-crystallization method, where powders of Li_2CO_3, B_2O_3, M_2O_3, and $LiCl$ were mixed in a $3 : 7(1-y) : 7y : 3.2$ ratio and melted and then quenched.

For a y-value of $3/7$, the complete replacement of BO_4 with MO_4 was expected to occur. They were able to determine that for a y-value of around 0.43 when using AlO_4, the solid solution formed from mixing and heating the starting powders contained a thermodynamically favorable combination of $Li_4B_7O_{12}Cl$ and $Li_4Al_3B_4O_{12}Cl$. They determined via powder x-ray diffraction analysis of the synthesis products that both

the substituted materials as well as their non substituted counterpart all had $F\bar{4}4c$ cubic structures. The structure determined from this work is reproduced in Figure 1.6.

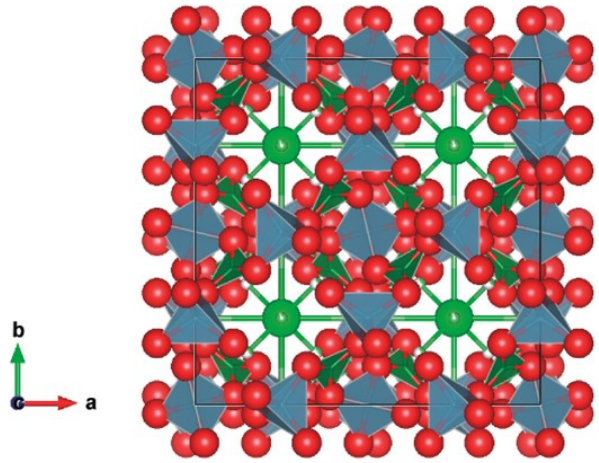


Figure 1.6: Figure showing the conventional cell of the disordered cubic $F\bar{4}3c$ structure of $Li_4Al_3B_4O_{12}Cl$ as reproduced from reference [19] with permission. Here the red, large green, and small green spheres denote O, Cl, and Li, and the green triangles and grey tetrahedra denote BO_3 and MO_4 units.

Furthermore, they were able to experimentally measure the conductivity of the aluminum and gallium substituted materials to both be about $1 \times 10^{-5}\ S\ cm^{-1}$ at room temperature, an order of magnitude higher than earlier works analyzing glass-ceramic and single crystal samples of the original boracite without boron substitution. Their conductivity results can be seen reproduced in Figure 1.7.

Although later works by Tan et al. [17] were able to produce a highly pure sample of $Li_4B_7O_{12}Cl$ with even higher conductivity on the order of $10^{-4}\ S\ cm^{-1}$, it seems that this simply suggests that synthesis and purity play a large role in the measured conductivity of the materials, and we believe that both the original boracite as well as the aluminum substituted counterpart are both good candidates for study as solid-state electrolytes.

We also reference a newer study from 2022 by Katsumata et al. [18] of aluminum

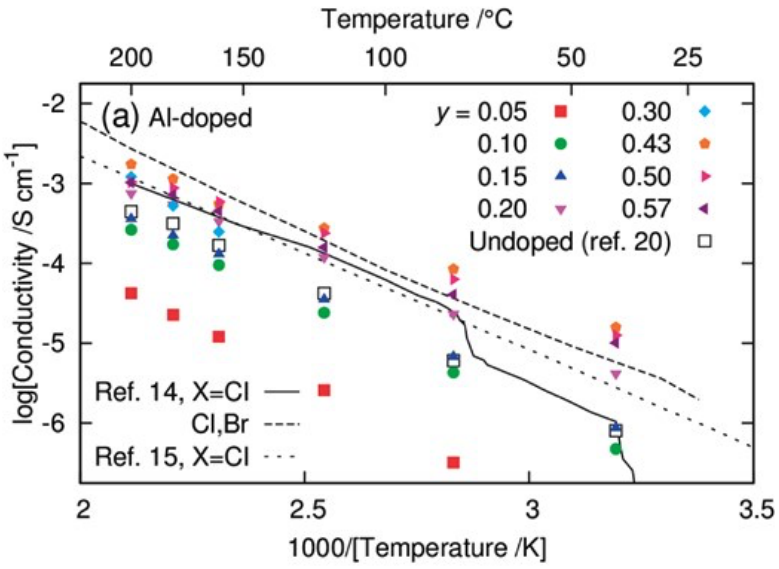


Figure 1.7: Figure showing Arrhenius plots of the ionic conductivity of $Li_4Al_3B_4O_{12}Cl$ as reproduced from reference [19] with permission. Here the different colored datapoints represent different doping ratios based on the choice of y during the synthesis procedure.

substitution of the lithium boracite alongside a new boracite with both aluminum and bromine partial substitution, $Li_4Al_3B_4O_{12}Cl_{1-x}Br_x$, which shows that the lithium boracties and various analogues with partial substitutions are still of current interest in research and continue to be promising solid-state electrolyte candidates.

1.2.3 Lithium Thioboracite – $Li_4B_7S_{13}I$

In addition to the lithium boracite and its derivatives, work has also been done to investigate thioboracite materials, which use sulfur in place of oxygen. From chemical intuition, we expect that sulfur may provide better performance but may be more reactive and thus have issues relating to chemical stability. A recent study by Kaup et al. [20] reported the results from their investigation into a related thioboracite material, $Li_6B_7S_{13}I$, with iodine as the halogen and higher number of lithium ions per unit cell than the oxygen boracite counterpart.

In this work they synthesized $Li_6B_7S_{13}I$ through an annealing process involving a closely related material composed of $Li_{7.5}B_{10}S_{18}I_{1.5}$. The process resulted in the formation of a crystalline phase with cubic $F\bar{4}3c$ symmetry as determined by single crystal x-ray diffraction. Upon further investigation involving the variation of synthesis conditions, they were able to subtly distort the crystalline structure to create a tetragonal polymorph of $Li_6B_7S_{13}I$ with $I\bar{4}c2$ symmetry. These two structures are reproduced in Figure 1.8.

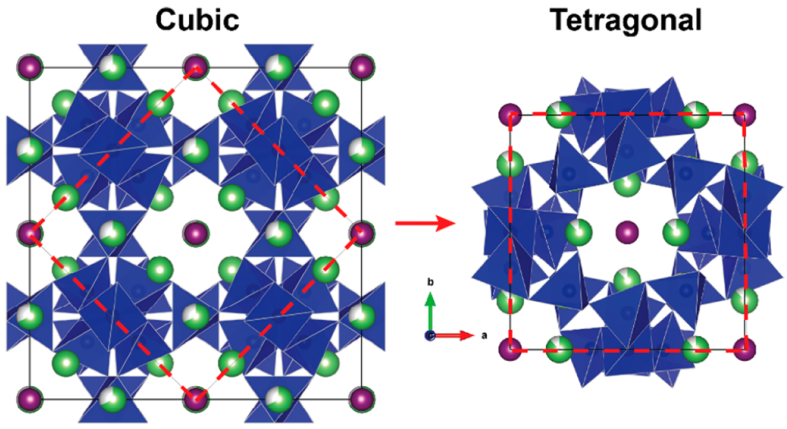


Figure 1.8: Figure showing the cell of the cubic $F\bar{4}3c$ and tetragonal $I\bar{4}c2$ structures of $Li_6B_7S_{13}I$ as reproduced from reference [20] with permission. Here the purple and green spheres represent I and Li, and the blue tetrahedra represent BS_4 units.

Analysis of this tetragonal phase indicated that it was dynamically unstable as indicated by imaginary phonon modes and that the cubic phase was preferred, as seen by convex hull analysis. They were also able to determine that the cubic phase has a very high theoretical ionic conductivity of 5.2×10^{-3} $S\ cm^{-1}$, making it an ideal candidate for solid-state electrolyte application. This can be seen from the Arrhenius plots reproduced in Figure 1.9. They further suggested that although the material exhibits some issues with stability, nanolayer coatings can further stabilize the material at the electrode interfaces.

This work suggests that thioboracites may additionally be promising candidates

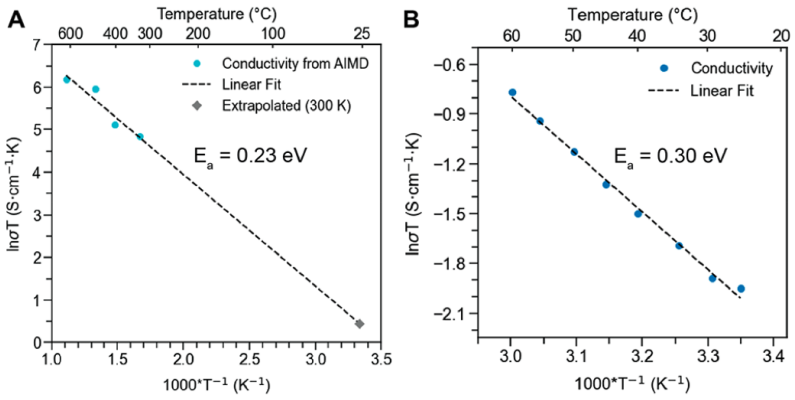


Figure 1.9: Figure showing Arrhenius plots of the ionic conductivity of $Li_6B_7S_{13}I$ as reproduced from reference [20] with permission. The figure on the left shows the data from first principles simulations and the figure on the right shows data from EIS measurements.

of study alongside the previously considered boracite materials. As a result, we have chosen to focus our works on the family of related boracite and thioboracite materials which encompasses the permutation space of the substitutions previously mentioned, namely the partial substitution of boron sites with aluminum, and the full substitution of oxygen sites with sulfur, resulting in eight possible materials of interest: $Li_4B_7O_{12}Cl$, $Li_4Al_3B_4O_{12}Cl$, $Li_4B_7S_{12}Cl$, $Li_4Al_3B_4S_{12}Cl$, $Li_6B_7O_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, $Li_6B_7S_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$. We note that only two of the boracite materials has been directly synthesized and investigated, one thioboracite material is derived from an experimentally synthesized and studied material with a different halogen (I instead of Cl), and the remaining five materials are considered purely from a theoretical standpoint.

1.3 Lithium Phosphate Solid Electrolytes

This section gives some details of the lithium phosphate - lithium orthoscilicate alloys as solid-state electrolytes. These alloys are well known, and serve as a suitable "test system" for our work investigating the feasibility of using the Allegro machine learned

interatomic potential to extend first principles methods as discussed in section 5.2. In our work, we do not utilize these alloys for material investigations.

1.3.1 Lithium Phosphate/Silicate Alloy - $Li_{3.25}P_{0.75}Si_{0.25}O_4$

Lithium phosphate (Li_3PO_4) has been studied for quite some time, as evidenced by research into the structure of the material done by Keffer et al. in 1967 [21]. This is a well known electrolyte material which has also been used in combination with lithium orthosilicate (Li_4SiO_4) to form solid-state electrolyte alloys as early as 1977, when research was done by Hu et al. investigating the conductivity of alloys containing various ratios of the two compounds [22].

More recently, works by Deng et al. [23] investigated various stochiometries of the Li_3PO_4/Li_4SiO_4 alloy and found the most conductive to be $(Li_3PO_4)_{0.75}(Li_4SiO_4)_{0.25}$. The unit cell for this structure was reported with fractional occupancy sites based on experimental and computational methods, which is shown in Figure 1.10.

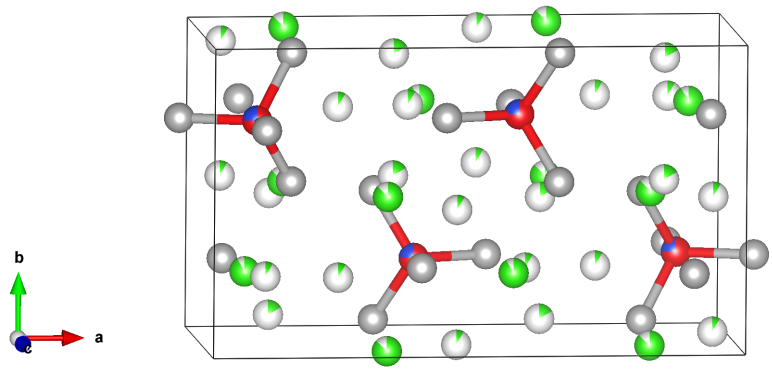


Figure 1.10: Figure showing the structure of $Li_3PO_4 + Li_4SiO_4$ based on the cif file as published by Deng et al. [23]. Here the green, blue, red, and grey coloring of the spheres represent lithium, silicon, phosphorous, and oxygen. The lithium sites are fractionally occupied and the phosphorous/silicon sites are occupied at a ratio of 3:1.

They were also able to find the conductivity to be quite high, around $10^{-3}\ S\ cm^{-1}$ at 573 K. MSD results from molecular dynamics was presented in their work, compar-

ing the lithium ion mobility over time for the various alloys with different stochiometries, which is reproduced in Figure 1.11.

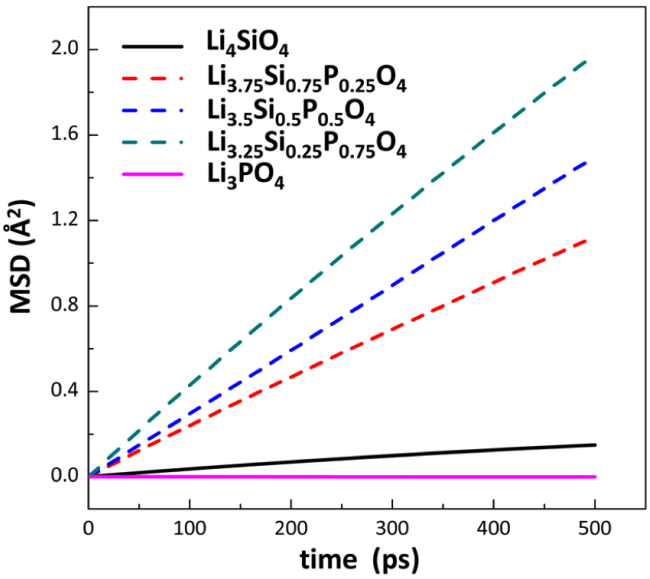


Figure 1.11: Figure showing MSD plots of the lithium ions of various alloys of Li_3PO_4 + Li_4SiO_4 as reproduced from reference [23] with permission.

We note that their ionic conductivity calculations were performed using emperical potentials ran for long simulation time with quite large simulation cells, which is not directly comparable to our utilization of this material for our test system as described in section 5.2. The structures we analyzed were only loosely based on the unit cell from the work by Deng et al. [23].

1.4 Lithium Argyrodite Solid Electrolytes

In this section, we briefly introduce two argyrodite materials which we investigated in collaboration with experimental work performed by Zachary D. Hood, a researcher at Argonne National Laboratory, and Udochukwu "Udo" Eze, a Georgia Tech graduate student. Our contributions to this work are discussed in section 5.4.

1.4.1 Lithium Argyrodite - Li_6PS_5Cl

The lithium argyrodite materials, named after their similarities to the naturally occurring argyrodite mineral Ag_8GeS_6, have been of great interest to researchers working on solid-state battery materials. They have cubic structures with very high symmetry that are characterized by space group $F\bar{4}3m(216)$ and have been found to have very high lithium ion conductivity. A study of the Li_6PS_5X argyrodite by Deiseroth et al. in 2008 [24] experimentally investigated this material with differing choices of halogens $X = Cl, Br, I$.

The structure consists of a framework composed of $[PS_4]^{-3}$ tetrahedra with Li^{+1}, X^{-1}, and S^{-2} ions located in symmetry sites within the framework. This study primarily investigated structural details using NMR spectroscopy, comparing the argyrodite analogues to one another. Figure 1.12 reproduces the crystalline structure of Li_6PS_5I as depicted in their work.

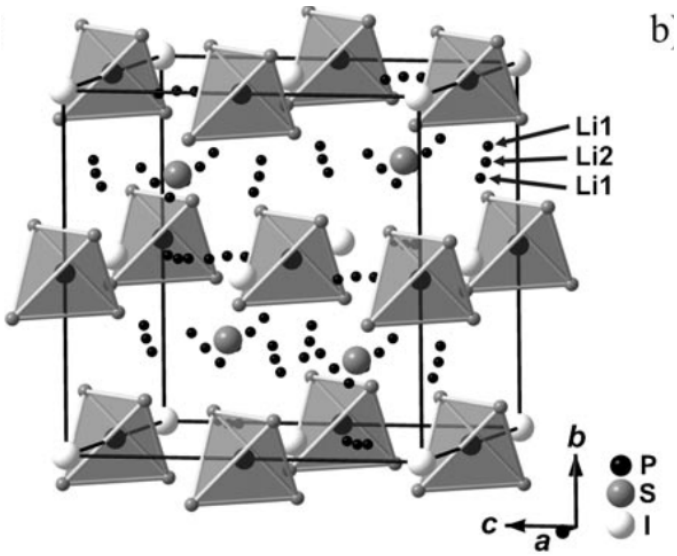


Figure 1.12: Figure showing a section of the crystalline structure of Li_6PS_5I as reproduced from reference [24] with permission.

A followup to this work by Deiseroth et al. in 2011 [25] continued investigation of

these three argyrodite materials while introducing a fourth argyrodite material with different stochiometry, Li_7PS_6. This paper primarily investigated the lithium diffusion pathways through these materials and performed experimental measurements of their ionic conductivity using impedence spectroscopy. An arrhenius plot depicting the conductivity of a Li_6PS_5Cl material sample is reproduced in Figure 1.13.

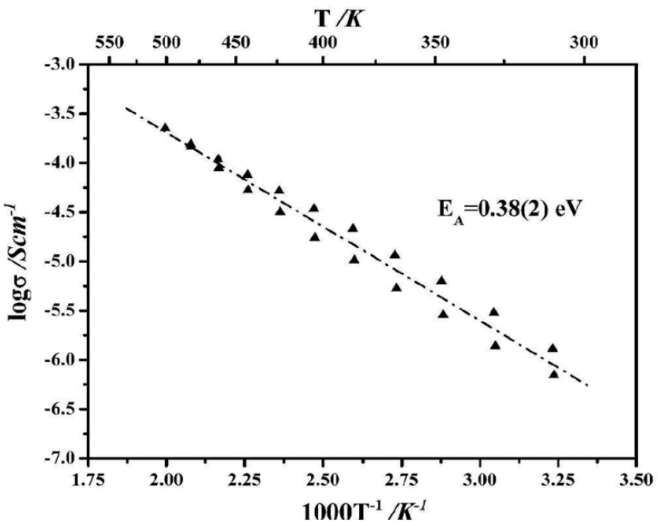


Figure 1.13: Figure showing an Arrhenius plot of Li_6PS_5Cl derived from impedence spectroscopy at 309K as reproduced from reference [25] with permission.

Ionic conductivities were reported as high as 10^{-6} $S\ cm^{-1}$ at low temperature (160K - 290K) and 10^{-4} $S\ cm^{-1}$ at high temperature (313K - 572K), demonstrating the high ionic conductivity of the material.

1.4.2 ZnO-doped Lithium Argyrodite - $Li_{5.5}PS_{4.75}Cl\,[ZnO]_{0.25}$

ZnO doping of the lithium argyrodite material has been investigated as a method of improving the arygrodite's stability in contact with electrodes in a solid-state battery. Common oxide based materials have high reactivity with the arygrodite, which results in unwanted degradation of the material. Additionally, a common issue when interfacing the argyrodite material with pure lithum is the formation of lithium den-

drites, which over time damage the battery and cause failure. A recent paper by Chen et al. in 2019 [26] has shown that by doping the Li_6PS_5X $(X = I, Br, Cl)$ with ZnO, the stability of the electrolyte in contact with the electrodes has greatly improved, reducing lithium dendrite formation while retaining the good lithium ionic conductivity required for this application.

Recent unpublished experimental work done by Dr. Zach Hood and Udo Eze at Argonne National Laboratory involved synthesis of ZnO doped Li_6PS_5Cl to further study the material. Their results were quite encouraging, and they showed good stability and performance of their newly synthesized material $Li_{5.5}PS_{4.75}Cl[ZnO]_{0.25}$. Our group was involved with an unpublished computational analysis of the structure of the ZnO doped argyrodite, which is presented in section 5.4.

Chapter 2: Select First Principles Methods

In this chapter, we introduce key theory which serves as a foundation for our first principles calculations. As we are investigating solid-state electrolytes, we need a way to model a periodically repeating crystalline lattice of atoms quantum mechanically, which involves methods of solving Schroedinger's equation for our system. This requires density functional theory and Bloch theory, and allows us to calculate the internal static lattice energy, atomic forces, and stress tensor of a crystalline solid quite accurately. We also desire to allow for vibrational degrees of freedom, which would allow us to calculate the phonon band structure and phonon density of states at zero temperature for the material, giving us an idea of dynamic stability of the material. This can be done using density functional perturbation theory or finite differences methods. Finally, we also consider perhaps the most important measure with which to assess suitability of a material for use as a solid-state battery electrolyte, the ionic conductivity. This involves running molecular dynamics calculations for long time to simulate ionic motion through the material as a function of time, and then using various transport equations and statistical methods to determine the ionic conductivity.

2.1 Quantum Theory of Crystals

Imagine we want to find the energy of some crystalline material of interest. We model this material as a unit cell which contains many atoms in fixed positions, typically with some symmetry considerations taking place, where the unit cell infinitely repeats in all spatial directions. To simplify this problem, let's first consider only a single unit cell - a fixed collection of atoms in space. This simpler system contains nuclei

with positions $\mathbf{R}_\alpha$ and electrons with positions $\mathbf{r}_i$. The nuclei have some mass m_α and charge q_α, which may be different if the nuclei are different atomic species or isotopes. The electrons all have the same fixed mass m_e and the same fixed charge q_e.

$$q_\alpha = Z_\alpha e \qquad q_e = -e \tag{2.1}$$

The nuclear and electronics charges can be written in terms of their atomic number Z_α and the elementary charge e as seen in equation 2.1. Now let's look at the Born Oppenheimer approximation [27] which allows us to simplify this model further, the details of which are in part informed by David Tong's "Quantum Mechanics" textbook [28] and Born and Haung's "Dynamical Theory of Crystal Lattices" textbook [29].

2.1.1 Born Oppenheimer Approximation

Given the known quantities and the composition of the atomic system from the nuclear and electronic coordinates, we can write down the exact Hamiltonian for the system as seen in equation 2.2, where T_N is the total nuclear kinetic energy, T_E is the total electronic kinetic energy, and V_{NN}, V_{EE}, and V_{NE} are the nucleus-nucleus, electron-electron, and nucleus-electron coulombic interaction potentials. The individual terms which comprise the Hamiltonian are defined in equations 2.3, 2.4, 2.5, 2.6, and 2.7.

$$H = T_N + T_E + V_{NN} + V_{EE} + V_{NE} \tag{2.2}$$

$$T_N = -\frac{\hbar^2}{2}\sum_\alpha \frac{1}{m_\alpha}\nabla_\alpha^2 \tag{2.3}$$

$$T_E = -\frac{\hbar^2}{2m_e}\sum_i \nabla_i^2 \tag{2.4}$$

$$V_{NN} = \frac{1}{2}\left[\frac{e^2}{4\pi\epsilon_0}\sum_{\alpha\neq\beta}\frac{Z_\alpha Z_\beta}{|\mathbf{R}_\alpha - \mathbf{R}_\beta|}\right] \tag{2.5}$$

$$V_{EE} = \frac{1}{2}\left[\frac{e^2}{4\pi\epsilon_0}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}\right] \tag{2.6}$$

$$V_{NE} = \frac{e^2}{4\pi\epsilon_0}\sum_{\alpha,i}\frac{-Z_\alpha}{|\mathbf{R}_\alpha - \mathbf{r}_i|} \tag{2.7}$$

Note that the factor of $1/2$ in equations 2.5 and 2.6 is present to avoid double counting of the particle pairs, whereas in equation 2.7, they are necessarily distinct and we need not divide by two. Now that we have the Hamiltonian, we can trivially write the exact time independent Schroedinger equation for the system as seen in equation 2.8.

$$H\Psi = E\Psi = \left[T_N + T_E + V_{NN} + V_{EE} + V_{NE}\right]\Psi \tag{2.8}$$

We immediately see that this equation is not possible to solve analytically, and we wish to simplify it to enable us to solve the problem approximately. Fortunately, as Born and Oppenheimer showed in their work, we can reduce the complexity significantly with a few assumptions.

For the first step of the Born-Oppenheimer approximation, we simply analyze the kinetic energy terms in isolation. If we rewrite T_N in terms of the atomic weight M_α, we have the kinetic energy term shown in equation 2.9 replacing equation 2.4.

$$T_N = -\frac{\hbar^2}{2u}\sum_\alpha \frac{1}{M_\alpha}\nabla_\alpha^2 \tag{2.9}$$

Here we have introduced the unified atomic mass unit u. If we compare the magnitudes of the prefactors of T_N and T_E, as seen in equation 2.10 below, we see that

the nuclear kinetic energy terms are much smaller in magnitude than the electronic kinetic energy terms.

$$\frac{\hbar^2}{2u} \approx \frac{\hbar^2}{2 \cdot 2000 m_e} << \frac{\hbar^2}{2m_e} \tag{2.10}$$

It then follows that we can approximate the nuclear kinetic energy to be exactly zero, i.e. $T_N \approx 0$. We are then left with the modified time independent Schroedinger equation seen in equation 2.11, where we have the approximate Hamiltonian $\hat{H}$ with total energy $\hat{E}$.

$$\hat{H}\Psi = [T_E + V_{NN} + V_{EE} + V_{NE}]\,\Psi = \hat{E}\Psi \tag{2.11}$$

This is often called the clamped-nuclei approximation. Next we consider how to solve this equation. We can separate the equation and define the approximate time independent electronic Schroedinger equation as seen in equation 2.12 below.:

$$H_E\phi_n = [T_E + V_{EE} + V_{NE}]\,\phi_n = \mathcal{E}_n\phi_n \tag{2.12}$$

where ϕ_n is the instantaneous electronic wavefunction in state n. We can solve this equation (somehow) for the electronic wavefunctions and energies, but we must remember we have assumed that the nuclei are stationary, therefore it is implied that $\phi_n = \phi_n(\{\mathbf{r}_i\};\{\mathbf{R}_\alpha\})$. In principle we can solve this for a large set of different nuclear positions to define an approximate potential energy surface for the electrons as a function of both $\{\mathbf{r}_i\}$ and $\{\mathbf{R}_\alpha\}$.

For the second step of the Born-Oppenheimer approximation, we revisit the full time independent Schroedinger equation 2.8. If we assume that the energy states of the system are non-degenerate, we can express the full solution as a sum over products of nuclear states ψ_n and electronic states ϕ_n, as seen in equation 2.13.

$$\Psi = \sum_n \psi_n(\{\mathbf{R}_\alpha\})\phi_n(\{\mathbf{r}_i\}) \tag{2.13}$$

For convenience, we can rewrite equation 2.8 in terms of the electronic Hamiltonian, H_E, as seen in equation 2.14. Here we introduce the nuclear Hamiltonian, $H_N = T_N + V_{NN}$.

$$H\Psi = [H_N + H_E]\,\Psi = E\Psi \tag{2.14}$$

For ease of notation, we switch to expressing only the electronic states in bra-ket notation. If we now take an inner product with a particular electronic state $\langle\phi_m|$, we end up with the expression seen in equation 2.15. We can then expand the expression in terms of $|\Psi\rangle = \sum_n \psi_n |\phi_n\rangle$, as seen in equation 2.16. This is called "integrating out" the electronic positions.

$$\langle\phi_m| H |\Psi\rangle = \langle\phi_m| H_N |\Psi\rangle + \langle\phi_m| H_E |\Psi\rangle = \langle\phi_m| E |\Psi\rangle \tag{2.15}$$

$$\langle\phi_m| H |\Psi\rangle = \sum_n \left(\langle\phi_m| H_N\psi_n |\phi_n\rangle + \langle\phi_m| H_E\psi_n |\phi_n\rangle\right) = \sum_n \langle\phi_m| E\psi_n |\phi_n\rangle \tag{2.16}$$

We can deal with each inner product sum separately, where we utilize the fact that $\langle\phi_m|\phi_n\rangle$ is zero if $m \neq n$ and one if $m = n$. We first deal with the easiest two of the three evaluations, shown in equation 2.17.

$$\sum_n \langle\phi_m| E\psi_n |\phi_n\rangle = E\sum_n \psi_n \langle\phi_m|\phi_n\rangle = E\psi_m \tag{2.17}$$

The second most straightforward evaluation starts in equation 2.18. We then operate on $|\phi_n\rangle$ with H_E, where $H_E |\phi_n\rangle = \mathcal{E}_n |\phi_n\rangle$ and we get equation 2.19.

$$\sum_n \langle\phi_m| H_E\psi_n |\phi_n\rangle = \sum_n \psi_n \langle\phi_m| H_E |\phi_n\rangle \tag{2.18}$$

$$\sum_n \psi_n \langle\phi_m| \mathcal{E}_n |\phi_n\rangle = \sum_n \mathcal{E}_n\psi_n \langle\phi_m|\phi_n\rangle = \mathcal{E}_m\psi_m \tag{2.19}$$

Finally we look at the most complex inner product, which does not easily reduce due to the fact that H_N acts on both ψ_n and ϕ_n, since they are both functions of the nuclear positions and H_N contains the gradient of the nuclear positions. We utilize the fact that $H_N = T_N + V_{NN}$ to expand the expression, which is shown in equation 2.20.

$$\sum_n \langle\phi_m| H_N\psi_n |\phi_n\rangle = \sum_n \langle\phi_m| T_N\psi_n |\phi_n\rangle + \sum_n \langle\phi_m| V_{NN}\psi_n |\phi_n\rangle \tag{2.20}$$

Since V_{NN} is independent of the electronic positions, we can easily evaluate this inner product as shown in equation 2.21.

$$\sum_n \langle\phi_m| V_{NN}\psi_n |\phi_n\rangle = V_{NN}\sum_n \psi_n \langle\phi_m|\phi_n\rangle = V_{NN}\psi_m \tag{2.21}$$

We are now left with one final inner product to evaluate, which is shown in equation 2.22.

$$\sum_n \langle\phi_m| T_N\psi_n |\phi_n\rangle = -\frac{\hbar^2}{2}\sum_n\sum_\alpha \langle\phi_m| \frac{1}{m_\alpha}\nabla_\alpha^2\psi_n |\phi_n\rangle \tag{2.22}$$

Using the vector identity $\nabla^2(AB) = A\nabla^2 B + B\nabla^2 A + 2(\nabla A \cdot \nabla B)$, we can rewrite this as seen in equation 2.23.

$$-\frac{\hbar^2}{2}\sum_n\sum_\alpha \langle\phi_m|\phi_n\rangle \frac{1}{m_\alpha}\nabla_\alpha^2\psi_n + C_1 + C_2 = -\frac{\hbar^2}{2}\sum_\alpha \frac{1}{m_\alpha}\nabla_\alpha^2\psi_m + C_1 + C_2 \quad (2.23)$$

Here we have introduced two nuclear-electronic coupling terms C_1 and C_2. if we assume this coupling is negligible, which is the case in the adiabatic approximation, we can neglect these terms and we now have the expression shown in equation 2.24.

$$-\frac{\hbar^2}{2}\sum_\alpha \frac{1}{m_\alpha}\nabla_\alpha^2\psi_m + V_{NN}\psi_m + \mathcal{E}_m\psi_m = E\psi_m \quad (2.24)$$

we can then define an effective Hamiltonian for the nuclei as seen in equation 2.25.

$$H_n^{\text{eff}} = -\frac{\hbar^2}{2}\sum_\alpha \frac{1}{m_\alpha}\nabla_\alpha^2 + V_{NN} + \mathcal{E}_n \quad (2.25)$$

Here we note that the effective nuclear Hamiltonian depends on the particular electronic state $|\phi_n\rangle$. For now, we will focus on the problem of solving for the electronic wavefunctions ϕ_n in the following sections.

2.1.2 Density Functional Theory

In this section we reference the works of Hohenberg and Kohn [30] and Kohn and Sham [31], which establish the foundations of density functional theory as a computational model. We also utilize information from Martin's "Electronic Structure" textbook [32] for developing the notation and derivation of the relevant equations. We start by recalling equation 2.12, which we reproduce here in equation 2.26.

$$H_E\phi_n = \left[T_E + V_{EE} + V_{NE}\right]\phi_n = \mathcal{E}_n\phi_n \quad (2.26)$$

Although we have made good progress thanks to Born and Oppenheimer in significantly reducing the complexity by trivializing the nuclear components, solving a many body system of electrons in a fixed electrostatic potential is still quite the task. Thanks to the brilliant work by Hohenberg and Kohn, we know that all properties of a quantum many-body system of particles can be considered unique functionals of the ground-state particle density, and we have a way forward which allows us to continue in our attempts to solve equation 2.26.

Formally, the Hohenberg and Kohn theory is based on two theorems, listed below.

1. For a system of interacting particles in some external potential, denoted $V_{\text{ext}}(\mathbf{r})$, the potential is determined uniquely up to a constant by the ground-state particle density $n_0(\mathbf{r})$.

2. A universal functional for the energy, denoted $E[n]$, can be defined in terms of the particle density $n(\mathbf{r})$, which is valid for any external potential $V_{\text{ext}}(\mathbf{r})$. For a particular choice of $V_{\text{ext}}(\mathbf{r})$, the exact ground-state energy of the system is equal to the global minimum value of this functional, and the specific particle density which minimizes this functional is the exact ground-state particle density $n_0(\mathbf{r})$.

What this means is that we can now reformulate our previous Hamiltonian given in equation 2.26 with the following Hamiltonian in equation 2.27 which can then be analyzed with the Hohenberg and Kohn theorems, where in our case $V_{\text{ext}}(\mathbf{r}_i)$ is the interaction energy between the nuclear charges and our i-th electron. For convenience we also change notation to denote the electronic energy E_E as simply E and the electronic Hamiltonian H_E as $\hat{H}$.

$$\hat{H}\phi = \left(-\frac{\hbar^2}{2m_e} \sum_i \nabla_i^2 + \sum_i V_{\text{ext}}(\mathbf{r}_i) + \sum_{i \neq j} \frac{e^2}{|\mathbf{r}_i - \mathbf{r}_j|} \right) \phi = E\phi \tag{2.27}$$

These two theorems are quite extraordinary, which of course require evidence to be believed. Fortunately, the proofs are relatively simple, and are provided below.

PROOF OF THEOREM I

This will be a simple proof by contradiction. Assume that there exist two unique external potentials $V^1_{\text{ext}}(\mathbf{r})$ and $V^2_{\text{ext}}(\mathbf{r})$ that differ by more than a constant yet give rise to the same ground-state density $n_0(\mathbf{r})$. It follows that this results in two unique Hamiltonians $\hat{H}^1$ and $\hat{H}^2$, each of which have different ground-state wavefunctions ϕ^1_0 and ϕ^2_0. Since $\phi^1_0 \neq \phi^2_0$, it must be the case that the inequality given in equation 2.28 holds.

$$E^1 = \langle \phi^1_0 | \, \hat{H}^1 \, | \phi^1_0 \rangle < \langle \phi^2_0 | \, \hat{H}^1 \, | \phi^2_0 \rangle \tag{2.28}$$

Note that this assumes that the ground-state in this case is non-degenerate. We can rewrite the last term in equation 2.28 as seen in equation 2.29.

$$\langle \phi^2_0 | \, \hat{H}^1 \, | \phi^2_0 \rangle = \langle \phi^2_0 | \, \hat{H}^2 \, | \phi^2_0 \rangle + \langle \phi^2_0 | \, \hat{H}^1 - \hat{H}^2 \, | \phi^2_0 \rangle = E^2 + \langle \phi^2_0 | \, \hat{H}^1 - \hat{H}^2 \, | \phi^2_0 \rangle \tag{2.29}$$

This leads to the inequality shown in equation 2.30.

$$E^1 < E^2 + \langle \phi^2_0 | \, \hat{H}^1 - \hat{H}^2 \, | \phi^2_0 \rangle \tag{2.30}$$

Similarly, we can start from the other direction, writing a similar inequality as shown in equation 2.31.

$$E^2 = \langle \phi^2_0 | \, \hat{H}^2 \, | \phi^2_0 \rangle < \langle \phi^1_0 | \, \hat{H}^2 \, | \phi^1_0 \rangle = \langle \phi^1_0 | \, \hat{H}^1 \, | \phi^1_0 \rangle + \langle \phi^1_0 | \, \hat{H}^2 - \hat{H}^1 \, | \phi^1_0 \rangle \tag{2.31}$$

This leads to the analogous inequality shown in equation 2.32.

$$E^2 < E^1 + \langle \phi_0^1 | \, \hat{H}^2 - \hat{H}^1 \, | \phi_0^1 \rangle \tag{2.32}$$

If we add equations 2.30 and 2.32, we get equation 2.33.

$$E^1 + E^2 < E^1 + E^2 + \langle \phi_0^2 | \, \hat{H}^1 - \hat{H}^2 \, | \phi_0^2 \rangle + \langle \phi_0^1 | \, \hat{H}^2 - \hat{H}^1 \, | \phi_0^1 \rangle \tag{2.33}$$

Since the ground-state densities for both ϕ_0^1 and ϕ_0^2 are $n_0(\mathbf{r})$, we find that $\langle \phi_0^2 | \, \hat{H}^1 - \hat{H}^2 \, | \phi_0^2 \rangle = - \langle \phi_0^1 | \, \hat{H}^2 - \hat{H}^1 \, | \phi_0^1 \rangle$ and equation 2.32 becomes $E^1 + E^2 < E^1 + E^2$, which is a contradiction. Therefore, it must be the case that there does not exist two unique external potentials $V_{\text{ext}}^1(\mathbf{r})$ and $V_{\text{ext}}^2(\mathbf{r})$ that differ by more than a constant yet give rise to the same ground-state density $n_0(\mathbf{r})$.

PROOF OF THEOREM II

If in general we consider the space of ground-state particle densities $n_0(\mathbf{r})$ which correspond to some external potential $V_{\text{ext}}(\mathbf{r})$, then we are able to create functionals within this space. Since we know that all properties of the system are uniquely determined given a specific $n_0(\mathbf{r})$ based on theorem I, then we can define the total energy functional as seen in equation 2.34.

$$E_{\text{HK}}[n] = T[n] + E_{\text{int}}[n] + \int d^3 r V_{\text{ext}}(\mathbf{r}) n_0(\mathbf{r}) + E_{NN} \tag{2.34}$$

Here we have introduced the term $E_{\text{int}}[n]$ which is the electron-electron interaction energy functional, analagous to equation 2.6. We also note that E_{NN} is the interaction energy of the nuclei, which is given in equation 2.5. Furthermore, we can rewrite equation 2.34 as seen in equation 2.35, where we have introduced the Hohenberg-Kohn functional $F_{\text{HK}} \equiv T[n] + E_{\text{int}}[n]$.

$$E_{\mathrm{HK}}[n] = F_{\mathrm{HK}} + \int d^3r V_{\mathrm{ext}}(\mathbf{r}) n_0(\mathbf{r}) + E_{NN} \tag{2.35}$$

We note that the functional F_{HK} must be universal by construction because the terms that define it are functionals of only the electron density, and their is no dependence on the nuclear configuration.

Now if we look at a specific system with ground-state electron density $n_0^1(\mathbf{r})$ corresponding to an external potential $V_{\mathrm{ext}}^1(\mathbf{r})$, we know from theorem I that the ground-state energy can be written as seen in equation 2.36.

$$E^1 = E_{\mathrm{HK}}[n_0^1(\mathbf{r})] = \langle \phi_0^1 | \, \hat{H}^1 \, | \phi_0^1 \rangle \tag{2.36}$$

If we now examine a different density, which we will call $n_0^2(\mathbf{r})$. Note that this can be any density at all in our large space of ground-state densities so long as it is not $n_0^1(\mathbf{r})$. This density similarly corresponds to the ground-state wavefunction $|\phi_0^2\rangle$, and we note that it must be the case that this density yields an energy higher than E^1, as show in equation 2.37.

$$E^1 = \langle \phi_0^1 | \, \hat{H}^1 \, | \phi_0^1 \rangle < \langle \phi_0^2 | \, \hat{H}^1 \, | \phi_0^2 \rangle = E^2 \tag{2.37}$$

It then follows that the energy given by evaluating the functional in equation 2.35 at the ground-state density $n_0^1(\mathbf{r})$, i.e. $E_{\mathrm{HK}}[n_0^1(\mathbf{r})]$, must be lower than it's evaluation at any other density. Since we have expressed this energy functional $E_{\mathrm{HK}}[n]$ in terms of a universal functional $F_{\mathrm{HK[n]}}$, it must also be universal, and furthermore if we knew F_{HK} exactly, we could minimize $E_{\mathrm{HK}}[n]$ in order to find the ground-state density and energy.

Now that we have definitively established the validity of the Hohenberg and Kohn theory, we wish to apply it to our problem given in equation 2.27. According to the theory, we know that our potential $V_{\text{ext}}(\mathbf{r})$ is determined uniquely (up to a constant) by the ground-state particle density, $n_0(\mathbf{r})$. We also know there exists a universal functional $E[n]$ such that $E[n_0(\mathbf{r})] = E_0$, where E_0 is the ground-state energy of the system. If we knew the functional exactly, we could solve for the ground-state energy and density by finding its global minimum. Unfortunately for us, it is impossible to find the functional, and we must employ other means to solve the problem.

Thankfully, Kohn and Sham devised a solution to this issue, which is to replace the complex many body system described by the Hamiltonian in equation 2.2 entirely with a much simpler auxiliary system. This is done based on the assumption that this auxiliary system has a ground-state density that is exactly equal to the ground-state density of the original system it replaces. Thanks to the Hohenberg Kohn theory, we know that if we solve this auxiliary system and find its ground-state density, then we know everything we could want to know about the system (and by extension, our original system, since the ground-state densities are the same). This auxiliary system is chosen intelligently to be a system with particles that are completely non-interacting, which means we only have to deal with solving a single particle Hamiltonian for each particle.

One may ask the obvious question, 'if we are solving single particle Hamiltonians for each particle that constitutes our non-interacting auxiliary system, how are the many-body interactions encoded?' The ingenuity of the Kohn Sham formalism is that they are neatly swept under the rug, being hidden in the exhange-correlational functional of the particle density. If we knew the exact form of this functional, we could use Kohn Sham theory to find exact solutions for our system. While we are guaranteed that this exact exchange-correlation exists, thanks to Hohenburg Kohn

theory, unfortunately existence of a functional is not correlated with possession of said functional. Unless we have an incredible amount of foresight or luck, we have no idea what the "true" exchange-correlation functional is.

Fortunately for us, there are many excellent approximate forms that we can use rather than attempting to divine the "true" functional out of thin air. The focus of many bodies of work surrounding density functional theory involve careful construction of physically meaningful approximations of the exchange-correlation functional, including Local Density Approximation [33], Perdew–Burke–Ernzerhof (PBE) [34], and many others. In the works presented in this dissertation, we exclusively use the modified PBE functional known as "PBEsol" [35]. As it is beyond the scope of this work, we forego further discussion of exchange-correlation functionals more specifically, but we do wish to present the Kohn Sham equations more thoroughly here.

For the auxiliary system, we can define the Hamiltonian as follows:

$$\hat{H}^{\sigma}_{\text{aux}} = -\frac{1}{2}\nabla^2 + V^{\sigma}(\mathbf{r}) \tag{2.38}$$

where we have adopted Hartree atomic units for simplicity such that $\hbar = m_e = 4\pi/\epsilon_0 = 1$. V^{σ} is defined to be a local potential acting on an electron of spin σ at located at $\mathbf{r}$. For a system with N independent electrons which are governed by this Hamiltonian, there are $N^{\uparrow}$ spin up electrons and $N^{\downarrow}$ spin down electrons, such that $N^{\uparrow} + N^{\downarrow} = N$. The ground-state of this system must therefore have one electron in each of the N^{σ} orbitals $\phi^{\sigma}_i(\mathbf{r})$ with the lowest eigenvalues of the Hamiltonian, ϵ^{σ}_i. The electron density can be written as follows:

$$n(\mathbf{r}) = \sum_{\sigma} n(\mathbf{r}, \sigma) = \sum_{\sigma} \sum_{i=1}^{N^{\sigma}} |\phi^{\sigma}_i(\mathbf{r})|^2 \tag{2.39}$$

Kohn and Sham showed that one can rewrite the Hohenberg Kohn expression for the ground-state energy functional, given in equation 2.35, as seen in equation 2.40 below:

$$E_{KS} = T_S[n] + \int d\mathbf{r} V_{\text{ext}}(\mathbf{r}) n(\mathbf{r}) + E_{\text{Hartree}} + E_{NN} + E_{xc}[n] \tag{2.40}$$

where we have defined several terms: $T_S[n]$ is the independent particle kinetic energy, V_{ext} is the external potential due to the nuclei and possibly other external fields, E_{Hartree} is the Hatree energy term, describing the electron density self interaction, V_{NN} is the interaction between the nuclei, and $E_{xc}[n]$ is the exchange correlation term. We now define T_S in equation 2.41, $E_{\text{Hartree}}[n]$ in equation 2.42, and $E_{xc}[n]$ in equation 2.43:

$$T_S = -\frac{1}{2} \sum_{\sigma} \sum_{i=1}^{N^\sigma} \langle \phi_i^\sigma | \nabla^2 | \phi_i^\sigma \rangle = \frac{1}{2} \sum_{\sigma} \sum_{i=1}^{N^\sigma} \int d^3 r |\nabla \phi_i^\sigma(\mathbf{r})|^2 \tag{2.41}$$

$$E_{\text{Hartree}}[n] = \frac{1}{2} \int d^3 r d^3 r' \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \tag{2.42}$$

$$E_{xc}[n] = F_{\text{HK}}[n] - (T_S[n] + E_{\text{Hartree}}[n]) \tag{2.43}$$

where $F_{\text{HK}}[n]$ is the Hohenberg Kohn functional defined earlier in this section. We note that without any other eternal fields, V_{ext} is simply the external potential due to the nuclei given in equation 2.7 and E_{NN} is again the nuclear-nuclear interaction term given by equation 2.5.

After a bit more work not presented here in the interests of time, they were able to arrive at the famous Kohn-Sham equations, given in equation 2.44 below:

$$\left(H_{\text{KS}}^\sigma - \epsilon_i^\sigma\right) \phi_i^\sigma = 0 \tag{2.44}$$

where ϵ_i^σ are the eigenvalues and H_{KS} is the effective Hamiltonian, which is given in Hartree atomic units in equation 2.45 below.

$$H_{\text{KS}}^\sigma(\mathbf{r}) = -\frac{1}{2}\nabla^2 + V_{\text{KS}}^\sigma(\mathbf{r}) \tag{2.45}$$

where $V_{\text{KS}}^\sigma(\mathbf{r})$ is defined in equation 2.46 below.

$$V_{\text{KS}}^\sigma(\mathbf{r}) = E_{\text{ext}}(\mathbf{r}) + V_{\text{Hartree}}(\mathbf{r}) + V_{\text{xc}}^\sigma(\mathbf{r}) \tag{2.46}$$

To solve the Kohn-Sham equations, we simply iteratively refine an initial guess at the electron density until the solution to the Kohn-Sham equations for the electron density is self consistent (up to convergence tolerances). In this work we utilize QUANTUM ESPRESSO (QE) [36] [37] [38] for performing these so-called self consistent field (SCF) calculations.

2.1.3 Hellmann Feynmann Theory and Interatomic Forces

Once we have solved the Kohn-Sham equations and have our ground-state energy in hand, we can utilize Hellmann-Feynman theory [cite] to determine the coulombic forces acting on a particular nucleus at position $\mathbf{R}_\alpha$. First we analyze the more general case where we want to determine the derivative of the total energy E with respect to some general parameter λ. We assume nothing about the particular system here, just that it has some energy eigenstates associated with eigen wavefunctions, i.e. $H\left|\Psi\right\rangle = E\left|\Psi\right\rangle$. We can then easily express E as the expectation value of the Hamiltonian, i.e. $E = \left\langle\Psi\right| H \left|\Psi\right\rangle$. If we now take the derivative of E and by applying the chain rule, we get the expression shown in equation 2.47.

$$\frac{\partial E}{\partial \lambda} = \langle\frac{\partial \Psi}{\partial \lambda}| H |\Psi\rangle + \langle\Psi| \frac{\partial H}{\partial \lambda} |\Psi\rangle + \langle\Psi| H |\frac{\partial \Psi}{\partial \lambda}\rangle \tag{2.47}$$

We note that the first and last terms in equation 2.47 can be simplified by operating with the Hamiltonian on the wavefunction $|\Psi\rangle$, resulting in the expression given in equation 2.48.

$$\frac{\partial E}{\partial \lambda} = E \langle \frac{\partial \Psi}{\partial \lambda} | \Psi \rangle + \langle \Psi | \frac{\partial H}{\partial \lambda} | \Psi \rangle + E \langle \Psi | \frac{\partial \Psi}{\partial \lambda} \rangle \tag{2.48}$$

Due to that fact that the derivative operation is commutative, for both the first and last term we can "factor out" the derivative operator to get the expression in equation 2.49.

$$\frac{\partial E}{\partial \lambda} = E \frac{\partial}{\partial \lambda} \langle \Psi | \Psi \rangle + \langle \Psi | \frac{\partial H}{\partial \lambda} | \Psi \rangle + E \frac{\partial}{\partial \lambda} \langle \Psi | \Psi \rangle \tag{2.49}$$

Since $\langle \Psi | \Psi \rangle = 1$, the first and last terms vanish after taking their derivative and we are left with the expression in equation 2.50.

$$\frac{\partial E}{\partial \lambda} = \langle \Psi | \frac{\partial H}{\partial \lambda} | \Psi \rangle = \langle \frac{\partial H}{\partial \lambda} \rangle \tag{2.50}$$

This equation is the general form of the Hellmann Feynmann theory. We now wish to apply it to our problem in order to determine the forces acting on a nucleus from the surrounding nuclei and electrons. Once we have solved the electronic problem for the total electronic energy E as a function of the nuclear positions, we can express the force of a given nucleus at position $\mathbf{R}_\alpha$ as seen in equation 2.51.

$$\mathbf{F}_\alpha = -\frac{\partial E}{\partial \mathbf{R}_\alpha} - \frac{\partial V_{NN}}{\partial \mathbf{R}_\alpha} \tag{2.51}$$

According to Hellmann Feynman theory, we can express the first term in equation 2.51 as seen in equation 2.52.

$$\frac{\partial E}{\partial \mathbf{R}_\alpha} = \langle \phi | \frac{\partial \hat{H}}{\partial \mathbf{R}_\alpha} | \phi \rangle \tag{2.52}$$

As a result of the application of the Hellmann Feynmann theory, we can express the force acting on the α-th nucleus at position $\mathbf{R}_\alpha$ as seen in equation 2.53.

$$\mathbf{F}_\alpha = -\int d^3 r n(\mathbf{r}) \frac{\partial V_{\text{ext}}(\mathbf{r})}{\partial \mathbf{R}_\alpha} - \frac{\partial V_{NN}}{\partial \mathbf{R}_\alpha} \tag{2.53}$$

2.1.4 Periodic Lattices and Bloch Theory

Up until now, we have have only been defining our system as a collection of nuclei and electrons. For solid-state materials, we know these materials form crystalline lattices with repeating unit cells that have very intricate and beautiful spatial symmetries. In this section we describe some of the details regarding the treatment of periodic systems using Bloch Theory [39], where the discussion is informed by several sources such as the "Solid State Physics" textbook by Ashcroft and Mermin [40] and Martin's "Electronic Structure" textbook [32].

We start by introducing some notation regarding periodically repeating crystalline lattices. Crystalline lattices are described by the concept known as the Bravias lattice. The Bravais lattice is a set of discrete points in 3 dimensional space of the form given in equation 2.54 below.

$$\mathbf{R} = n_1 \mathbf{a_1} + n_2 \mathbf{a_2} + n_3 \mathbf{a_3} \tag{2.54}$$

Here we have introduced the primitive lattice vectors $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$, which span $\mathbb{R}^3$, and integers n_1, n_2, and n_3. We can see an illustration of this construction in Figure 2.1 below.

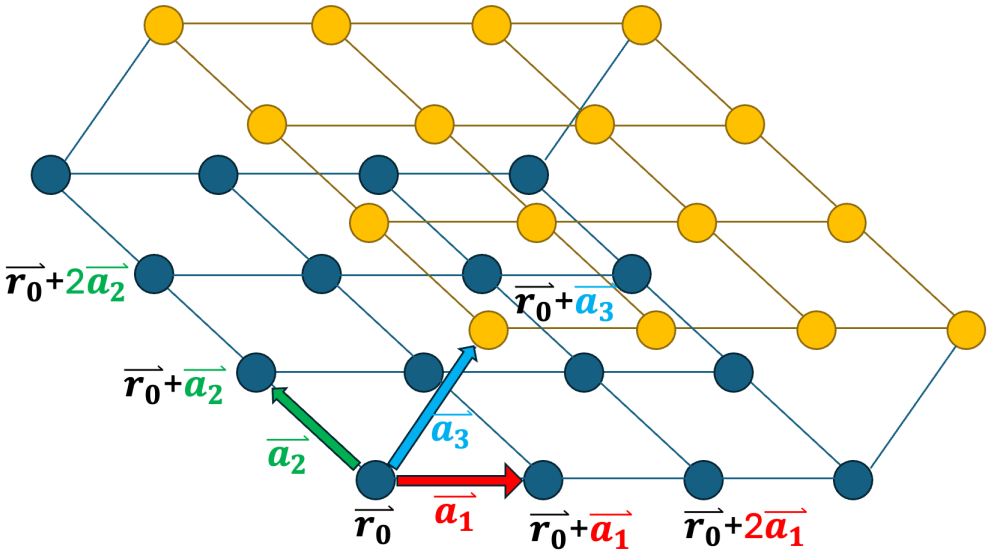


Figure 2.1: Figure depicting a Bravais lattice with primitive lattice vectors $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$.

Within the context of Bravais lattices, one can define a unit cell, which is simply a volume of space that when translated by the subset of vectors of the form given in equation 2.54, it fills the larger space without overlapping or leaving voids. A **primitive cell** is simply the smallest of such unit cells, which contains a single lattice point and can be represented as seen in equation 2.55 below.

$$\mathbf{r} = x_1\mathbf{a_1} + x_2\mathbf{a_2} + x_3\mathbf{a_3} \tag{2.55}$$

Here we have defined the **fractional coordinates** x_1, x_2, and x_3, where each term is a real number between 0 and 1, i.e. $0 \leq x_i \leq 1$. A **conventional cell** is a larger cell comprised of several primitive cells, which is selected based on the symmetry of the crystal and often defined to be a cubic cell. The conventional cell has lattice vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ which are different than the primitive lattice vectors. One can also imagine constructing even larger cells, sometimes called **supercells** of the crystalline

material.

Crystalline materials also have beautiful symmetries that can be classified by a particular **space group**. A space group is simply a description of a repeating pattern in space, which defines how atomic sites in a crystalline material are related by translations, rotations, inversions, and reflections. There are a total of 230 unique space groups which define crystallographic materials. **Wyckoff positions** are special points contained within the space group that are symmetrically equivalent, and are denoted by a special letter which labels positions of a specific type. More general discussion of crystallography including definitions of the space groups and their unique sets of Wyckoff labels can be found in an introductory textbook on the subject such as "Crystal Structure Determination" by Massa [41]. Below we show an example of a crystalline form of $NaCl$, which is described by the face-centered cubic $Fm\bar{3}m$ space group, seen in Figure 2.2 below.

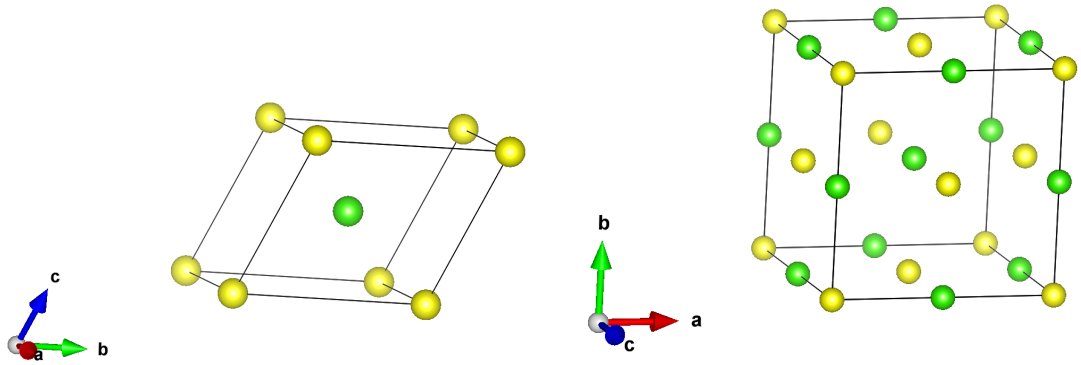


Figure 2.2: Figure depicting the primitive (left) and conventional cell (right) of $NaCl$ in the face centered cubic crystalline form described by the $Fm\bar{3}m$ space group. The yellow and green spheres represent sodium and chlorine. The cif files used to prepare these figures were obtained through the materials project database [42].

We find that it has Wyckoff sites in the primitive and conventional cells as listed below in Table 2.1.

Here we see that in the primitive cell, Na atoms are located at lattice points with

Table 2.1: Table depicting the Wyckoff positions of the primitive and conventional cells of $NaCl$ described by the $Fm\bar{3}m$ space group, based on information from the materials project database [42].

Primitive Cell					Conventional Cell				
Label	Element	a	b	c	Label	Element	a	b	c
1a	Na	0	0	0	4a	Na	0	0	0
1b	Cl	1/2	1/2	1/2	4b	Cl	0	0	1/2

Wyckoff label $1a$, where the letter a denotes the special symmetry site itself, and the number 1 means there is only 1 such site in the primitive cell. Similarly, in the conventional cell, the Wyckoff label $4a$ means that there are 4 special sites of type a where Na are located. These special sites are given in fractional coordinates (a, b, c) of the lattice vectors, where the additional sites of that type (where applicable) can be found by performing the appropriate space group symmetry operations. It is important to note that in the case of the conventional cell the 4 sites of type a or b are symmetrically equivalent.

By analyzing these space groups and special symmetry sites known as Wyckoff positions, we can analyze ground-state structures to see what kind of symmetries are present as well as use symmetry based arguments to search for new structures, such as finding ordered structures from disordered reference structures or to find suitable candidate structures where atomic substitution is performed.

In addition to defining primitive cells in real space as shown in equation 2.54, we can also define the reciprocal lattice, which is itself a Bravais lattice of a special form defined in reciprocal space, as shown below in equation 2.56.

$$\mathbf{K} = k_1\mathbf{b_1} + k_2\mathbf{b_2} + k_3\mathbf{b_3} \tag{2.56}$$

Here we are considering the set of all wave vectors $\mathbf{K}$ which define plane waves with the periodicity of the Bravais lattice defined by $\mathbf{R}$, meaning they must satisfy equation

2.57 below.

$$e^{i\mathbf{K}\cdot\mathbf{R}} = 1 \tag{2.57}$$

The reciprocal lattice vectors can be written in terms of the primitive lattice vectors as seen in equations 2.58, 2.59, and 2.60,

$$\mathbf{b}_1 = 2\pi \frac{\mathbf{a}_2 \times \mathbf{a}_3}{\mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)} \tag{2.58}$$

$$\mathbf{b}_2 = 2\pi \frac{\mathbf{a}_3 \times \mathbf{a}_1}{\mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)} \tag{2.59}$$

$$\mathbf{b}_3 = 2\pi \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)} \tag{2.60}$$

where it must be the case that $\mathbf{b}_i \cdot \mathbf{a}_j = 2\pi\delta_{ij}$. Furthermore, we can define the **first Brillouin zone** to be the **Wigner-Seitz primitive cell** in reciprocal space. The Wigner-Seitz primitive cell is a special primitive cell defined for a Bravais lattice which has its single lattice point in the center and has its boundary defined by the set of points in space that are closer to the central lattice point than any other lattice point. The Brillouin zone is an important component of phonon band calculations.

Finally, we can briefly define **Bloch's Theorem** below:

- For a one-electron Hamiltonian $H = -\hbar^2\nabla^2/2m + U(\mathbf{r})$ where $U(\mathbf{r}+\mathbf{R}) = U(\mathbf{r})$ for all $\mathbf{R}$ in a Bravais lattice, the eigenstates ϕ can be chosen to have the form of a plane wave multiplied by a function with the periodicity of the Bravais lattice, as seen in equation 2.61.

$$\phi_{n\mathbf{k}}(\mathbf{r}) = e^{(i\mathbf{k}\cdot\mathbf{r})} u_{n\mathbf{k}}(\mathbf{r}) \tag{2.61}$$

Here $u_{n\mathbf{k}}(\mathbf{r}) = u_{n\mathbf{k}}(\mathbf{R}+\mathbf{r})$ must hold for all $\mathbf{R}$ in the Bravais lattice. Alternatively, we can restate this in the following way in equation 2.62:

$$\phi(\mathbf{r}+\mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{R}}\phi(\mathbf{r}) \tag{2.62}$$

where the wave functions are constructed in such as way that the equality holds true for all $\mathbf{R}$ in the Bravais lattice. The proof of this theorem and further discussion is omitted for brevity, but we note that this theorem is central to solving the Kohn-Sham equations for a periodically repeating lattice. This is done by constructing the Kohn-Sham orbitals which satisfy equation 2.44 to be of the form seen in equation 2.61. This means that our Kohn-Sham orbitals are defined to be Bloch waves, which can then be solved using SCF calculations.

2.1.5 Pseudopotentials

Electrons close to their nucleus have very strong electrostatic interactions, resulting in Bloch wavefunctions for the valence electrons that are complex and highly oscillatory. We know from chemistry that the "core" electrons do not participate in chemical boding and therefore do not change much. Were we to compute the valence electron wavefunctions for the real potential, we would have extraordinary processing cost for little actual effect on the interactions with neighboring atoms. To significantly decrease computational cost with little effect on accuracy, we adopt the "frozen core" model for these electrons near the nucleus, where they remain fixed in space and ignored, and the potential in this region for the valence electrons is replaced by a "pseudopotential", which is a smooth function that is easier to use in calculations. The region where the pseudopotential function replaces the real potential is defined by the cutoff radius r_{cut}, at which point the pseudopotential and real potential meet continuously and differentiably. Due to the interests of time, this section will not be

expanded upon here, but we note that there are many nice references where this topic is expanded on further, such as the original paper introducing the methodology by Hellmann [43], the Phillips-Kleinman (PK) pseudopotential method [44], Vanderbilt's paper on soft pseudopotetnials [45], and others, including Martin's "Electronic Structure" textbook [32]. In our calculations in the works presented in this dissertation, we use the **Projector Augmented Wave** formalism instead of simpler pseudopotentials, which improves the pseudopotential formalism by reconstructing the true electron wavefunctions for the core electrons using linear transformation operators applied to the smooth pseudo-wavefunctions. References for the PAW method include the original paper by Blochl [46] and the description given in Martin's "Electronic Structure" textbook [32]. In our works, we utilize PAW datasets in QE that were calculated with the ATOMPAW software package written by Holzwarth, Tackett, and Matthews [47].

2.2 Phonons in Crystals

Phonons are quasiparticles that represent quantized vibrational energies within a crystalline lattice. By finding solutions for phononic energy modes, we can determine the dispersion relation curves along a path through the first Brillouin zone, which can tell us a lot of interesting information about a material. Dynamic instability can be inferred from imaginary phonon modes, optical properties can be can be determined from high frequency phonon modes, acoustic properties can be determined from low frequency phonon modes, and generally the overall band structure can be used as a fingerprint for a material, allowing for the identification of a material's structure or for finding defects in a material. By calculating the phonon density of states, we can determine macroscopic properties of the material such as heat capacity, thermal conductivity, and the Helmholtz Free Energy of the material.

The phonons can be represented as harmonic perturbations to nuclei in the density functional theory framework, where they are accurately calculated using **Density Functional Perturbation Theory** (DFPT). This method has been used in the past extensively by our group. In the works presented in this dissertation, however, we utilize the **Finite Differences** (FD) method as implemented in the Phonopy software package developed by Togo and Tanaka [48] [49] [50], which is described in detail in this section. In our experience, the finite differences method gives nearly identical results to DFPT, while having significant computational speedup.

2.2.1 Finite Differences Method

As implemented in the Phonopy software package, phonons can be considered using the "frozen phonon" approach, where all calculations are done for static lattice structures. If the nuclei are treated classically, we can write down Newton's equation as seen below in equation 2.63:

$$M_I \frac{\partial^2 \mathbf{R}_I}{\partial t^2} = \mathbf{F}_I(\mathbf{R}_I) = -\frac{\partial}{\partial \mathbf{R}_I} E(\mathbf{R}) \tag{2.63}$$

Here we have introduced the mass M_I and position $\mathbf{R}_I$ of nucleus I and have defined $\mathbf{R}$ to be the set of all nuclear positions, i.e. $\mathbf{R} = \{\mathbf{R}_I\}$. $\mathbf{F}_I$ is the total force acting on nucleus I. Here we are assuming the Born Oppenheimer approximation is valid, therefore when defining the force $\mathbf{F}_I$, we only need consider coulombic forces between the nuclei and the electronic forces are determined by the electronic structure. The equilibrium positions of the nuclei are defined by $\mathbf{R}^0 = \{\mathbf{R}_I^0\}$ such that $\mathbf{F}_I(\mathbf{R}^0) = 0$. We can define the components of the force constant matrix within the harmonic approximation as seen in equation 2.64 below.

$$C_{I,\alpha;J,\beta} = \frac{\partial^2 E(\mathbf{R})}{\partial \mathbf{R}_{I,\alpha} \mathbf{R}_{J,\beta}} \tag{2.64}$$

where I and J indicate atomic indicies and α and β indicate cartesian coordinates. The vibrational modes of frequency ω within the harmonic approximation are described by the displacements seen in equation 2.65 below.

$$\mathbf{u}_I(t) = \mathbf{R}_I(t) - \mathbf{R}_I^0 = \mathbf{u}_I e^{i\omega t} \tag{2.65}$$

By plugging equations 2.64 and 2.65 into equation 2.63, we can rewrite the expression as follows in equation 2.66.

$$-\omega^2 M_I u_{I\alpha} = -\sum_{J,\beta} C_{I,\alpha;J,\beta}\ u_{J,\beta} \tag{2.66}$$

The full solution for all vibrational modes can then be expressed by the matrix equation as shown in equation 2.67 below.

$$\det \left| \frac{1}{\sqrt{M_I} M_J} C_{I,\alpha;J,\beta} - \omega^2 \right| = 0 \tag{2.67}$$

For a crytalline material, the atomic displacements must obey Bloch's theorem, where they can be defined for atom s in a given unit cell $\mathbf{T}_n$ of the form $\mathbf{T}_n = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3$, as seen in equation 2.68 below.

$$\mathbf{u}_s(\mathbf{T}_n) = e^{i\mathbf{k}\cdot\mathbf{T}_n} \mathbf{u}_s(\mathbf{k}) \tag{2.68}$$

By substituting equation 2.68 into equation 2.67, we can solve equation 2.69 below for the dispersion curves $\omega_{i,\mathbf{k}}$.

$$\det \left| \frac{1}{\sqrt{M_s M_{s'}}} C_{s,\alpha,s',\alpha'}(\mathbf{k}) - \omega_{i,\mathbf{k}}^2 \right| = 0 \tag{2.69}$$

where the so-called reduced force constant matrix elements $C_{s,\alpha,s',\alpha'}$ for wavenumber $\mathbf{k}$ are given by equation 2.70 below.

$$C_{s,\alpha,s',\alpha'}(\mathbf{k}) = \sum_{\mathbf{T}_n} e^{i\mathbf{k}\cdot\mathbf{T}_n} \frac{\partial^2 E(\mathbf{R})}{\partial \mathbf{R}_{s,\alpha}(0) \partial \mathbf{R}_{s',\alpha'}(\mathbf{T}_n)} = \frac{\partial^2 E(\mathbf{R})}{\partial \mathbf{u}_{s,\alpha}(\mathbf{k}) \partial \mathbf{u}_{s'\alpha'}(\mathbf{k})} \tag{2.70}$$

We note that to solve the set of equations defined by equation 2.69 as implemented in Phonopy, the displacements are approximated according to a Taylor series expansion and numerically solved, which is often called the "finite differences" method. While there are more details that could be elaborated on in this section, in the interests of time, we forego further detailed discussion. We note that by integrating the dispersion curves over the Brillouin zone, we can obtain the phonon density of states. We could also derive macroscopic thermodynamic properties such as heat capacity or free energy should we be interested in analysis of these quantities.

2.3 Transport Properties of Crystals

In this section, we briefly look at some of the theoretical concepts that underlie ionic conductivity calculations, which includes molecular dynamics simulations, the Green-Kubo formalism, and the tracer approximation.

2.3.1 Molecular Dynamics Simulations

Here we discuss the basic formalism behind molecular dynamics. We restrict our focus to molecular dynamics within the thermodynamic ensemble of constant volume, energy, and number of particles, NVE. To treat other ensembles with molecular

dynamics, we would need to implement one of the many approaches to regulate temperature and/or pressure, such as the Nose-Hoover thermostat [51] [52] [53] or the Berendsen barostat [54]. Some of the details in this section are informed by the contents of Frenkel and Smit's "Understanding Molecular Simulation" textbook [55] and Tuckermann's "Statistical Mechanics: Theory and Molecular Simulation" textbook [56].

In molecular simulations, the nuclei are treated as classical particles governed by Newton's second law. The one dimensional equation of motion we need to solve for a given particle's trajectory between time t_1 and t_2 is trivially defined in equation 2.71 below.

$$r(t) = \int_{t_1}^{t_2} v(t)dt \tag{2.71}$$

Since we cannot generally solve this equation analytically, we imagine integrating in very short steps along the trjectory so that we can approximate the integration numerically in an iterative manner. We simply start with a Taylor expansion of the position around time t, as seen in equation 2.72:

$$r(t + \Delta t) = \sum_{n=0}^{\infty} \frac{D^n[r(t)]}{n!}(\Delta t)^n \tag{2.72}$$

where $D^n[r(t)]$ is the n-th derivative of $r(t)$. If we expand the series up to a few terms, we have the following expression seen in equations 2.73 and 2.74:

$$r(t+\Delta t) = \frac{D^0[r(t)]}{0!}(\Delta t)^0 + \frac{D^1[r(t)]}{1!}(\Delta t)^1 + \frac{D^2[r(t)]}{2!}(\Delta t)^2 + \frac{D^3[r(t)]}{3!}(\Delta t)^3 + ... \tag{2.73}$$

$$r(t + \Delta t) = r(t) + v(t)(\Delta t) + \frac{a(t)}{2}(\Delta t)^2 + \frac{1}{3}\frac{da}{dt}(\Delta t)^3 + ... \tag{2.74}$$

Similarly, we have the Taylor expansion evaluated at $t - \Delta t$ as seen in equation 2.75:

$$r(t - \Delta t) = r(t) - v(t)(\Delta t) + \frac{a(t)}{2}(\Delta t)^2 - \frac{1}{3}\frac{da}{dt}(\Delta t)^3 + ... \tag{2.75}$$

Adding equations 2.74 and 2.75, we get equation 2.76 seen here:

$$r(t + \Delta t) + r(t - \Delta t) = 2r(t) + a(t)(\Delta t)^2 + ... \tag{2.76}$$

which can be rewritten and approximated by truncation of higher order terms as seen in equation 2.77:

$$r(t + \Delta t) \approx 2r(t) - r(t - \Delta t) + a(t)(\Delta t)^2 \tag{2.77}$$

We know we can simply substitute $a(t)$ as seen in equation 2.78 to end up with the final expression given in in equation 2.79.

$$a(t) = \frac{F(t)}{m} \tag{2.78}$$

$$r(t + \Delta t) \approx 2r(t) - r(t - \Delta t) + \frac{F(t)}{m}(\Delta t)^2 \tag{2.79}$$

This is known as the **Verlet Algorithm**, and is widely used in molecular dynamics software such as QUANTUM ESPRESSO and LAMMPS [57]. In QUANTUM ESPRESSO and ABINIT, being quantum mechanical codes, the forces acting on the nuclei are coulombic forces which are calculated using DFT. In LAMMPS, these forces are derived from emperical potentials or, in our case, machine learned potentials. We note that in the Verlet algorithm, the positions at time $t + \Delta t$ are not calculated from velocities, therefore if we wish to obtain the velocities, we must derive them by

subtracting equation 2.75 from equation 2.74, as seen below in equations 2.80 and 2.81.

$$r(t+\Delta t) - r(t-\Delta t) = 2v(t)(\Delta t) + \frac{2}{3}\frac{da(t)}{dt}(\Delta t)^3 + ... \tag{2.80}$$

$$v(t) = \frac{r(t+\Delta t) - r(t-\Delta t)}{2(\Delta t)} - \frac{1}{3}\frac{da(t)}{dt}(\Delta t)^4 - ... \tag{2.81}$$

If we again truncate the series we can approximate the velocity at time t as seen below in equation 2.82.

$$v(t) \approx \frac{r(t+\Delta t) - r(t-\Delta t)}{2(\Delta t)} \tag{2.82}$$

We note that due to the point of truncation in the approximations given in equations 2.79 and 2.82 that the error in the position is fairly low on the order of $(\Delta t)^4$, and the error in velocity is significantly higher, on the order of $(\Delta t)^2$. An improvement to the Verlet algorithm is the **Velocity Verlet algorithm**, where the velocities are explicitly updated every time step instead of being averaged as seen in equation 2.82, which is in fact the integration algorithm we use in our work.

Now that we have the positions and velocities in a form that can be numerically integrated, we note that before we can begin integration, we must define our initial positions and velocities. The positions are given as the equilibrium structural positions, and the velocities are randomly generated from some statistical distribution such as a Maxwell-Boltzmann distribution at some defined target temperature T, which is defined below in equation 2.83.

$$P(v_i) = \left(\frac{m_i}{2\pi k_B T}\right)^{3/2} e^{\left(\frac{m_i v_i^2}{2k_B T}\right)} \tag{2.83}$$

We note that there are other considerations not taken into account here, such as how the unit cell is setup, how the perodicity of the cell is taken into account, how the center of mass is corrected, etc. These details will be omitted for brevity. We do note that for our calculations, we utilize the Box-Muller equations [58] for generating our initial velocities.

2.3.2 Calculating Ionic Conductivity

We now presume we have used our favorite program and method for treating our intermolecular forces and have generated some particle trajectories. If these particles happen to be ionic, we can ask the question, 'what is the ionic conductivity of the material we are modeling?' To answer this question, we must utilize linear response theory to analyze our ionic trajectories. The ionic conductivity can be calculated as an equilibrium ensemble average over the ionic current density as expressed in the famous Green-Kubo formula [59] [60] [61] [62] given below in equation 2.84.

$$\sigma_{\mathrm{GK}} = \frac{\Omega}{3k_B T} \int_o^\infty \langle \mathbf{J}(t) \cdot \mathbf{J}(0) \rangle \, dt \tag{2.84}$$

Here we introduced the ionic current density at time t, $\mathbf{J}(t)$ and the system volume Ω. The equilibrium ensemble average $\langle \mathbf{J}(t) \cdot \mathbf{J}(0) \rangle$ is called the autocorrelation function of the ionic current density, which measures similarity between the current density at time t and at time 0. The beauty of this expression is that we are able to relate the macroscopic quantity σ_{GK} to the microscopic current densities. The ionic current density can be defined as seen below in equation 2.85.

$$\mathbf{J}(t) = \frac{1}{\Omega} \sum_{i=1}^{N_i} q_i \mathbf{v}_i(t) \tag{2.85}$$

where N_i is the number of ions in our system, q_i is the ion's charge, and $\mathbf{v}_i(t)$ is the ions velocity at time t. Following the works of Grasselli and Baroni [63] [64], it can be shown that this ionic charge can be expressed in terms of Born effective charges $\mathbf{Z}_i^*(t)$, which is a tensor whose components are defined below in equation 2.86.

$$Z^*_{i\alpha\beta} = \frac{\partial \mu_\alpha}{\partial r_{i\beta}} \tag{2.86}$$

where μ is the system's dipole such that $\mu = \Omega \mathbf{P}$ and $\mathbf{P}$ is the polarization such that $\dot{\mathbf{P}} = \mathbf{J}$. Fortunately, rather than deal with the more complex Born effective charges, it turns out the ionic charge q_i is well represented by the typical integer oxidation charge and we assume that q_i is defined as such from here onward.

It is often convenient to rewrite equation 2.84 in terms of the polarization density, by using the terms defined in equations 2.87 and 2.88 below.

$$\int_0^t \mathbf{J}(t')dt' = \mathbf{P}(t) - \mathbf{P}(0) \tag{2.87}$$

where $\mathbf{P}(t)$ is defined as seen in equation 2.88:

$$\mathbf{P}(t) = \frac{1}{\Omega} \sum_{i=1}^{N_i} q_i \mathbf{r}_i(t) \tag{2.88}$$

The revised σ_{GK} formula in terms of the polarization can found to be the expression given in equation 2.89 below.

$$\sigma_{\mathrm{GK}} = \frac{\Omega}{6k_B T} \lim_{\tau \to \infty} \frac{1}{\tau} \left\langle |\mathbf{P}(\tau) - \mathbf{P}(0)|^2 \right\rangle \tag{2.89}$$

Here we are averaging over an ensemble of systems with simulation time τ, which in practice is done by sampling time intervals of length τ from a relatively long

MD simulation. Historically, the Green-Kubo formalism has not been possible to accurately calculate due to the incredible noise present which requires averaging over a massive number of instances of a time interval τ to overcome the noise. This large number of time intervals τ necessarily requires very long MD runs which have previously been intractable.

As a result of these limitations, historically researchers have used the tracer approximation in place of the full conductivity as defined by the Green-Kubo formalism. The tracer approximation assumes that only the diffusing ions significantly contribute to the overall ionic conductivity and that the ionic correlations are negligible. The tracer conductivity can be calculated from the Mean Squared Displacement of the diffusing ions, as seen below in equation 2.90.

$$\sigma_{\text{tr}} = \frac{q_D^2}{6k_BT\Omega} \lim_{\tau\to\infty} \frac{1}{\tau}\text{MSD}(\tau) = \frac{q_D^2}{6k_BT\Omega} \lim_{\tau\to\infty} \frac{1}{\tau} \left\langle \sum_{i\in D} |\mathbf{r}_i(\tau) - r_i(0)|^2 \right\rangle \tag{2.90}$$

Here we have introduced the charge of the diffusing ions q_D, which again is the integer oxidation charge, and in the MSD function we are summing over all ions $i \in D$ where D is the set of diffusing ions in the system. Unfortunately, for many ionic conducting materials, the ionic correlations are not negligible, and the tracer approximation is not sufficient. In our preprint discussed in section 5.2, we investigate the possibility of using interatomic potentials to enable us to run the very long MD simulations necessary such that we can calculate the full conductivity without having to use the tracer approximation.

Chapter 3: Select Machine Learning Methods

In this chapter, I introduce the basic foundations of machine learning, and in particular, neural network definitions and constructions. The goal of this chapter is to provide sufficient knowledge with which to parse **Machine Learned Interatomic Potential (MLIP)** implementations in the next chapter. This material has been integral in the process of familiarizing myself with the underlying language and definitions in order to apply them to solving problems of interest in physics. Machine learning is a broad category of computational methods which are used to optimize a predictive function with respect to an input training set of examples. This includes many neural network architectures with wide ranging applicability, with implementations in fields ranging from language processing and image recognition models to the hard sciences including biology, chemistry, and physics. This also includes methods other than neural networks, including support vector machines and others, which we will not detail here. The information in this chapter has been written using my own notation which is derived from information obtained from various sources on neural networks [65] [66] [67] [68] [69]. We also borrow notation from mathematicians, where functions f are defined as mappings from vector space V to vector space W with the notation $f : V \to W$, and $\mathbb{R}^n$ is the vector space over the field of real numbers of dimension n.

3.1 Introduction to Neural Networks

Neural Networks are powerful machine learning tools that are being implemented across a broad spectrum of research areas to solve problems ranging from image recognition and natural language processing to protein folding and molecular simulations.

Neural Networks can be thought of as function approximators, which, when given a training dataset of known inputs and outputs, will learn the functional relationship between the inputs and outputs. Here we briefly introduce some important Neural Network concepts and definitions that are needed to understand the architectures behind these successful machine learning models.

3.1.1 Feed Forward Neural Networks

A basic **Feed Forward Neural Network (FFNN)**, also called a **Multi-Layer Perceptron (MLP)**, is a simple construction of linear functions of inputs combined with non-linear activation functions. The basic unit of a FFNN is the **artificial neuron**.

DEF: An **artificial neuron** η is defined as follows:

$$\boldsymbol{\eta : \mathbb{R}^n \rightarrow \mathbb{R}} \tag{3.1}$$

$$\eta(\mathbf{x}) = \sigma(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b) \tag{3.2}$$

where $\mathbf{x} \in \mathbb{R}^n$, $w_1, w_2, ..., w_n$ are learned **weights**, b is a learned **bias**, and σ is a **non-linear activation function**.

We see that we can express this equation in vector form as follows:

$$\eta(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b) \tag{3.3}$$

where

$$\mathbf{w}^T = \begin{bmatrix} w_1 & w_2 & ... & w_n \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3.4}$$

FFNNs are composed of **layers**, which are simply sets of unconnected artificial neurons operating in parallel. Each neuron in the layer has independent weights and biases, which are given by $w_{i,j}$ and b_i, where the i is the neuron index and j is the weight index for a given neuron.

DEF: A **FFNN layer** Γ containing m artificial neurons is defined as follows:

$$\boldsymbol{\Gamma : \mathbb{R}^n \rightarrow \mathbb{R}^m} \tag{3.5}$$

$$\Gamma(\mathbf{x}) = \begin{bmatrix} \eta_1(\mathbf{x}) \\ \eta_2(\mathbf{x}) \\ \vdots \\ \eta_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \sigma(w_{1,1}x_1 + w_{1,2}x_2 + ... + w_{1,n}x_n + b_1) \\ \sigma(w_{2,1}x_1 + w_{2,2}x_2 + ... + w_{2,n}x_n + b_2) \\ \vdots \\ \sigma(w_{m,1}x_1 + w_{m,2}x_2 + ... + w_{m,n}x_n + b_m) \end{bmatrix} \tag{3.6}$$

where $\mathbf{x} \in \mathbb{R}^n$. Again we see that we can express this equation in vector form as follows:

$\Gamma(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{B})$

where $\mathbf{W}$ is the **weight matrix** for the layer, $\mathbf{B}$ is the **bias matrix** for the layer, and σ is the layer's non-linear activation function, where it is implied to operate element-wise on the resultant vector from the linear operation, $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{B}$. See the definitions below:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & ... & w_{1,n} \\ w_{2,1} & w_{2,2} & ... & w_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m,1} & w_{m,2} & ... & w_{m,n} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \qquad \sigma(\mathbf{z}) = \begin{bmatrix} \sigma(z_1) \\ \sigma(z_2) \\ \vdots \\ \sigma(z_m) \end{bmatrix} \tag{3.7}$$

Finally, we can fully define a FFNN mathematically as a composition of k layers. Each layer contains different sets of neurons, with independent weights and biases which are given by $w_{i,j}^{[k]}$ and $b_i^{[k]}$, where the superscript k in brackets is the layer index, i is the neuron index, j is the weight index for a given neuron. Note that each layer can have a different number of neurons as well as different activation functions.

DEF: A fully connected **FFNN** Ω containing k layers is defined as follows:

$$\boldsymbol{\Omega : \mathbb{R}^n \rightarrow \mathbb{R}^m} \tag{3.8}$$

$$\Omega(\mathbf{x}) = \Gamma^{[k]}(\Gamma^{[k-1]}(...(\Gamma^{[2]}(\Gamma^{[1]}(\mathbf{x}))))) \tag{3.9}$$

where we have implied that the inputs and outputs of the layers are "compatible," i.e.: $\Gamma^{[q]} : \mathbb{R}^{N_{q-1}} \rightarrow \mathbb{R}^{N_q}$ where $N_0 \equiv n$ and $N_k \equiv m$, i.e.:

$$\boldsymbol{\Gamma^{[1]} : \mathbb{R}^n \rightarrow \mathbb{R}^{N_1}} \tag{3.10}$$

$$\boldsymbol{\Gamma^{[2]} : \mathbb{R}^{N_1} \rightarrow \mathbb{R}^{N_2}} \tag{3.11}$$

$$... \tag{3.12}$$

$$\boldsymbol{\Gamma^{[k-1]} : \mathbb{R}^{N_{k-2}} \rightarrow \mathbb{R}^{N_{k-1}}} \tag{3.13}$$

$$\boldsymbol{\Gamma^{[k]} : \mathbb{R}^{N_{k-1}} \rightarrow \mathbb{R}^m} \tag{3.14}$$

We can think of this general construction of Ω as simply an arbitrary mapping from $\mathbb{R}^n \rightarrow \mathbb{R}^m$, whose functional form is found by training the FNNN on a dataset of known inputs and their corresponding outputs. This training fits the function Ω to the data, and is clearly dependent on the FNNN's **architecture** - the specific details of the configuration of the FNNN (how many layers, neurons per layer, activation functions, etc.). This is done by first initializing the weights and biases of the network

(which will likely be a poor approximation of the mapping we are looking for), which are then updated through an optimization algorithm that iteratively improves the fit until "convergence" is achieved.

3.1.2 Activation Functions

Which activation functions to use are important choices to make when designing the FFNN architecture. The choice is made for each layer and it is often the case that the activation function is the same for all **hidden layers** - layers that are in between the input and output layers. The **input layer** is the first layer of the network where raw or preprocessed data is read and the inputs to the network $\mathbf{x}$ are constructed. The **output layer** is the final layer of the network where the predicted outputs for a given input are calculated, which is often a **linear layer** - meaning the activation function is simply: $\sigma(\mathbf{z}) = \mathbf{z}$.

The family of activation functions commonly used and the justifications for picking one over another for a given type of learning problem is an area that is well studied [70], but will be omitted here for brevity. Common activation functions used include the logistic function, logistic(x), hyperbolic tangent, tanh(x), rectified linear units, ReLU(x), and sigmoid linear units, SiLU(x). These four functions are shown in Figures 3.1 and 3.2 to illustrate their functional forms. The logistic function is well suited for classification tasks, outputting in the range of $(0, 1)$. The hyperbolic tangent is centered at zero with a range of $(-1, 1)$, and improves the efficiency of the optimization compared to the logistic function by having a steeper derivative. Both of these functions have issues with "vanishing gradients" during optimization, where the function's derivatives at a particular point are close to zero and the model stops learning too early. ReLU fixes this issue for positive inputs due to the fact that it's derivative is always 1, but can have issues for negative inputs where they are forced

to be 0, causing the gradient to abruptly vanish. ReLU has a range of $(0, \infty)$ and is incredibly efficient, but also has a derivative that is discontinuous, which can be an issue in physics contexts where the predictive function is often an energy and its derivative defines atomic forces. SiLU is a newer activation function which is similar to ReLU with a range of roughly $(-0.28, \infty)$ and is continuously differentiable, but is computationally more expensive. By allowing a small dip into the negative numbers, the function avoids the issues with negative inputs where the gradient abruptly becomes 0. It also avoids the issue of vanishing gradients from positive inputs. In our work with Allegro, we have chosen to use the SiLU function.

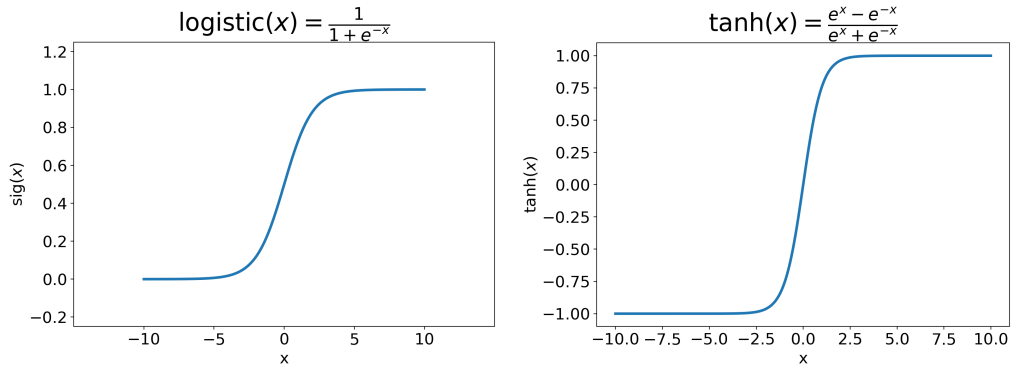


Figure 3.1: Plots of the logistic (left) and hyperbolic tangent (right) activation functions.

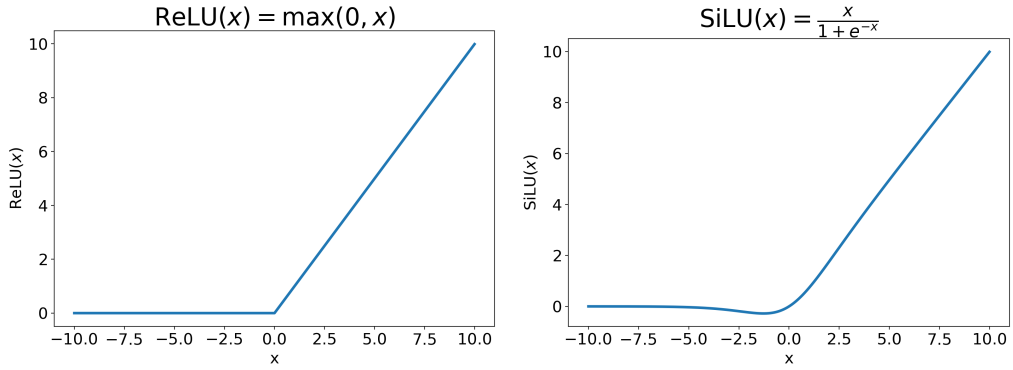


Figure 3.2: Plots of the ReLU (left) and SiLU (right) activation functions.

3.1.3 Loss Functions

Now that we have defined the predictive function Ω defining a FFNN, we need to understand how to initialize and optimize the FFNN based on a training dataset of known examples of inputs and their corresponding outputs. Weights and biases are often initialized to be between 0 and 1 and generated randomly through a statistical distribution, and inputs to each layer are typically normalized in some way to prevent the forward propagation through the network from diverging [71]. These details will not be discussed here in the interests of time, but instead we would like to focus on going over how the weights and biases are iteratively updated to optimize the network's predictive capabilities given a dataset of training examples. To do so we must define a function which measures the error of our predictive function, which we seek to minimize during optimization.

DEF: A **loss function** is a measure of error between the FFNN's predicted outputs given a set of inputs from a single training example and the actual outputs that are known from that training example.

Although there are several others, two common loss functions used in FFNNs are the **least squares** - also called **L2** - loss function and the **binary cross entropy (BCE)** loss function. The former is suitable for **regression** tasks, while the latter is used for **classification** tasks. Regression tasks are those for which we want to find a functional relationship between inputs and outputs, such as trying to find a Machine Learning Interatomic Potential (MLIP). Classification tasks, on the other hand, are those where you want to categorize data such that given a set of inputs, you predict a probability of that data belonging to a certain class. A classic example would be the task of image recognition, i.e. given an input image, we want to predict the probability that the image belongs to a given class in the set $\{cat, dog, person, bird, car, ...\}$.

The MLIP neural networks almost always use $L2$ loss functions, but we also define the binary cross entropy loss for completeness.

DEF: The **L2 loss function** for the $i-th$ example in a training dataset is defined as follows:

$$\mathcal{L}_{L2}^{(i)} = (\mathbf{y}^{(i)} - \Omega(\mathbf{x}^{(i)}))^2 \tag{3.15}$$

where the superscript enclosed in parenthesis denotes the training example index (is NOT an exponent), and the example is given by the set of inputs and outputs $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$

DEF: The **Binary Cross Entropy (BCE) loss function** for the $i-th$ example in a training dataset is defined as follows:

$$\mathcal{L}_{BCE}^{(i)} = -\mathbf{y}^{(i)} \log(\Omega(\mathbf{x}^{(i)})) - (1 - \mathbf{y}^{(i)}) \log(1 - \Omega(\mathbf{x}^{(i)})) \tag{3.16}$$

where again the superscript enclosed in parenthesis denotes the training example index (is NOT an exponent), and the example is given by the set of inputs and outputs $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$

Note that loss functions are not just arbitrary functions - in the case of the $L2$ and BCE loss functions, they are actually derivable from statistical analysis of the corresponding linear/logistic regression problems using maximum likelihood estimation, which is beyond the scope of this dissertation [72].

3.1.4 Optimization and Backpropagation

Now that we have our feed forward propagation equations and a suitable loss function to estimate the error for a given training example compared to the model's prediction, we want to optimize the model so that it can learn the function that best fits the

data. This is done by using an algorithm which updates the weights and biases in such as way that minimizes the loss function. One common algorithm is the **Gradient Descent Algorithm (GDA)**. We must first note that there are 2 different versions of the algorithm - **Batch Gradient Descent (BGD)** and **Stochastic Gradient Descent (SGD)**. SGD iterates over the training set randomly, calculating the loss for the iteration's training example and updating the weights and biases based on the error until "convergence" is achieved.

DEF: **Stochastic Gradient Descent (SGD)** is the algorithm defined as follows using C++ pseudo code:

```
int t = 1; //iteration index
int i = 0; //dataset index
while (!converged) {
  //generate a random training dataset index
  i = rand() % numTrainingExamples;
  //for each layer...
  for (int l = numLayers - 1; l >= 0; l--) {
    //for each neuron in the layer...
    for (int m = 0; m < numNeurons[l]; m++) {
      //for each weight in the neuron...
      for (int n = 0; n < numWeights[l]; n++) {
        //update the weight according to the equation
        w^{[l]}_{m,n}(t) = w^{[l]}_{m,n}(t-1) - \alpha \frac{\partial \mathcal{L}^{(i)}}{\partial w^{[l]}_{m,n}(t-1)}
      }
      //update the bias according to the equation
      b^{[l]}_{m}(t) = b^{[l]}_{m}(t-1) - \alpha \frac{\partial \mathcal{L}^{(i)}}{\partial b^{[l]}_{m}(t-1)}
    }
  }
  //check convergence
  if (checkEarlyStoppingCriteria()) {
    //stop looping
    converged = true;
  else {
    //increment t and continue
    t++
}
```

In this pseudo code, we have defined some self explanatory variables that will depend on the architectural choices and training dataset provided (numTrainingExamples,

numLayers, numNeurons, and numWeights). In the weight and bias update definitions, we have a variable α in front of the derivative of the loss function, which is called the **learning rate (LR)** - this value determines the magnitude of the adjustments being made to the weights and biases during update iterations, and is a user defined parameter. If α is too large, we will overshoot the minimum of $\mathcal{L}$ and can end up with oscillatory divergence. If α is too small on the other hand, we may end up learning too slowly and have convergence time trend towards infinity. The starting LR is highly problem-dependent, where common choices may range from $0.001 - 0.1$. We note that our definition here iterates randomly over the training set through selection with replacement. This means that some training examples may not be used and others may be used more frequently. In most implementations, the random selection is done without replacement, ensuring that the entire training dataset is utilized.

The LR can be managed by an **LR Scheduler**, whereby it is lowered (or raised) dynamically based on the loss function response over time to the training updates - for instance if we have reached a plateau and the loss function is no longer being reduced, we can lower the LR and "escape" the plateau to continue training and further improve the model. These details regarding schedulers are omitted for brevity. The "convergence" of the model is also a user defined parameter, and can be a set of multiple possible conditions - such as the loss reaching a defined minimum, the LR reaching a defined minimum through scheduling, or the time limit of the training being reached.

Though SGD, as one might imagine, is computationally efficient, it is quite noisy as the weights are being adjusted on a single training example at a time. BGD is a bit more costly in terms of computational resources, but is less noisy and improves the fit more generally at each iteration. Instead of using individual loss functions to update the weights and biases, the BGD uses a **cost function** - a sum of the individual

loss functions of the training examples contained in the training batch. The training batch can either be the entire dataset, which is called **Full Batch Gradient Descent (FBGD)**, or a subset of the dataset, which is called **Mini Batch Gradient Descent (MBGD)**. Often times the MBGD method is implemented by dividing up the dataset into randomly selected batches of a specified number of training examples, which are then iterated over to update the weights and biases. Unfortunately, the nomenclature is often a bit loose, and cost functions are often simply called loss functions, where there's a difference between "loss functions of a single example" and "loss functions of a batch of examples". In this dissertation, we will make the distinction by using the term "cost function". Oftentimes the cost function will be normalized by dividing by the number of examples in the batch.

DEF: The **normalized L2 cost function** for a batch of N_{batch} training examples can be defined as follows:

$$\mathcal{J}_{L2} = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \mathcal{L}_{L2}^{(i)} = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} (\mathbf{y}^{(i)} - \Omega(\mathbf{x}^{(i)}))^2 \tag{3.17}$$

where the superscript enclosed in parenthesis denotes the training example index (is NOT an exponent), and the examples are given by the set of their inputs and outputs $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$

DEF: **Batch Gradient Descent (BGD)** is the algorithm defined as follows using C++ pseudo code:

```
int t = 1; //iteration index
while (! converged) {
  //for each batch
  for (int i = 0; i < numBatches; i++) {
    //for each layer...
    for (int l = numLayers - 1; l >= 0; l--) {
```

```
      //for each neuron in the layer...
      for (int m = 0; m < numNeurons[l]; m++) {
        //for each weight in the neuron...
        for (int n = 0; n < numWeights[l]; n++) {
          //update the weight according to the equation
          w^[l]_{m,n}(t) = w^[l]_{m,n}(t-1) - α ∂J^(i) / ∂w^[l]_{m,n}(t-1)
        }
        //update the bias according to the equation
        b^[l]_m(t) = b^[l]_m(t-1) - α ∂J^(i) / ∂b^[l]_m(t-1)
      }
    }
  }
  //check convergence
  if (checkEarlyStoppingCriteria()) {
    //stop looping
    converged = true;
  else {
    //increment t and continue
    t++
}
```

In this pseudo code we have replaced the loss function for the $i-th$ training example with the cost function for the $i-th$ batch of training examples from the full dataset. If numBatches $= 1$, we are using the entire dataset as a single batch update, and thus are using FBGD, otherwise, we are using MBGD.

As a final note, we briefly discuss why this algorithm is iterated backwards through the network to update the weights and biases - starting at the final layer and iterating the the previous layers, hence the name **backpropagation**. We can easily see that for a given weight in the $q-th$ layer in a K layer FFNN, say $w^{[q]}_{1,2}$, we need to compute a lot of partial derivatives to find the derivative of the cost function with respect to this weight:

$$\frac{\partial \mathcal{J}}{\partial w^{[q]}_{1,2}} = \frac{\partial \mathcal{J}}{\partial \Gamma^{[K]}} \cdot \frac{\partial \Gamma^{[K]}}{\partial \Gamma^{[K-1]}} \cdot \ldots \cdot \frac{\partial \Gamma^{[q+1]}}{\partial \Gamma^{[q]}} \cdot \frac{\partial \Gamma^{[q]}}{\partial w^{[q]}_{1,2}} \tag{3.18}$$

If we instead start with the final layer and work our way backwards through the network, we need only compute the derivative of the next layer with respect to the

current layer at each iteration, as we will have all of the previous derivatives already computed. In our example, by the time we get to the $q-th$ layer in the backpropagation process, we will have already computed $\frac{\partial \mathcal{J}}{\partial \Gamma^{[K]}}, \frac{\partial \Gamma^{[K]}}{\partial \Gamma^{[K-1]}}, ..., \frac{\partial \Gamma^{[q+2]}}{\partial \Gamma^{[q+1]}}$ in the previous layers and only need to compute $\frac{\partial \Gamma^{[q+1]}}{\partial \Gamma^{[q]}} \frac{\partial \Gamma^{[q]}}{\partial w_{1,2}^{[q]}}$.

In addition to the gradient descent algorithms discussed here, there are several other optimization algorithms which have been researched, such as the **Conjugate Gradient Descent** [73]. There are also optimization algorithms which improve upon the typical gradient descent process by normalizing and averaging weights over time to make the learning "smoother", such as Adam [74] and AdamW [75]. AdamW in particular improves the Adam optimization algorithm by introducing weight decay, which shrinks large model weights and helps prevent overfitting. In our work with Allegro, we utilize the Adam optimization algorithm in the interests of time, based partially on the fact that AdamW requires extra fine-tuning of the weight decay factor, as well as the fact that Adam was the default optimization algorithm during our initial Allegro testing.

3.1.5 Input Features

Up until now, we have defined a general FFNN as a function acting on some input vector $\mathbf{x}$, which we now define as the **input feature vector** to the neural network. This feature vector is not just a simple set of inputs one might typically think of when constructing functions, i.e. a position vector $\overrightarrow{r} \in \mathbb{R}^3$, a set of charges $\{q_i\}$, and so on, or even a collection of these typical inputs. One might imagine building a FFNN to "learn" the coulomb potential for a set of n charges and naively think of constructing the input feature vector as follows:

$$\mathbf{x} = \begin{bmatrix} \overrightarrow{r_1} \\ q_1 \\ \overrightarrow{r_2} \\ q_2 \\ \vdots \\ \overrightarrow{r_n} \\ q_n \end{bmatrix} \tag{3.19}$$

While you can certainly define your input feature vector as the direct set of input variables, this is not a good idea in practice. The features are combined through linear transformations and non-linear activation functions that are very simple constructions which are used to build a predictive function in a piecewise manner. By only giving direct inputs, it will be very difficult to "learn" the correct functional form. We must instead intelligently construct features that are forms with which the neural network can use to build the correct function more easily. This is perhaps the most important part of defining your neural network such that it can effectively construct a suitable predictive function when trained.

3.1.6 Neural Network Flexibility and Limitations

As mentioned previously, FFNNs are essentially function approximators - given a training dataset and a loss function, they will "learn" a function that approximately maps the inputs to the outputs. But what is the limit of how good the approximation can be? Are there functions that can't be approximated well or at all by FFNNs? It turns out there are many **Universal Approximation Theorems (UATs)** for FFNNs - each with different assumptions about the FFNN and functions to be approximated - that tell us that, in fact, we can approximate any function at any arbitrary level of accuracy with FFNNs [76]. The UAT for a FFNN of arbitrary depth and constrained width can be defined as follows:

Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^n$, $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-linear continuously differentiable activation function, and $\mathcal{N}(\sigma; n, m; m+n+2)$ denote the function space of neural networks with n input neurons, m output neurons, and an arbitrary number of hidden layers with $m+n+2$ neurons, whose hidden neurons all have activation functions σ and whose output neurons all have activation functions $a(\mathbf{x}) = \mathbf{x}$. For any chosen function $f \in C(\mathcal{X}, \mathbb{R}^m)$ and constant $\epsilon > 0$ **there exists** a FFNN $\hat{f} \in \mathcal{N}(\sigma; n, m; m+n+2)$ such that $sup_{x \in \mathcal{X}} \|\hat{f}(x) - f(x)\| < \epsilon$. This means that for any arbitrary level of prediction error ϵ, there exists a FFNN of the general construction given such that for every choice of $x \in \mathcal{X}$, $\|\hat{f}(x) - f(x)\| < \epsilon$.

This is a very powerful statement with a few very obvious caveats. The knowledge that there exists a FFNN for any function you want to approximate with arbitrarily low error doesn't tell you **what** the FFNN architecture actually is or how to find it. In principle what this means is that FFNNs have the capability to approximate any given function and are therefore flexible enough to be used without worrying that it's not even possible to approximate the function you are trying to fit. In practice, larger numbers of neurons per layer and larger number of layers can improve the capacity of the network, and you still need to feed the FFNN a sufficiently large and diverse training set to optimize the weights and biases of the network to find the general functional form rather than just find a local function that fits the small sample data provided.

Given that FFNNs are so incredibly general and non-specific, as mentioned previously what seems to help the most is feature engineering - picking features that best represent your input data to "steer" the FFNN towards an appropriate solution. An example of this would be picking polynomials of increasing order as your features, so

that the FFNN can build a function with combinations of polynomial functions with your raw inputs as their arguments. This approach has utility, and has been used successfully in many cases, but a perhaps better approach is to add structure to the basic FFNN to improve its flexibility without painstakingly engineering features by hand.

3.2 Graph Neural Networks

One of the early extensions of the basic FFNN was the **Convolutional Neural Network (CNN)**. A CNN essentially replaces the input feature vector of a FFNN with an input matrix which is processed through so-called convolutional filters before being converted into a vector which then propagates through a FFNN as its input feature vector. The idea here is that matrices have a spatial structure, and convolutions can create localized features in space. CNNs are used heavily for image recognition tasks, and have also been used in physical applications as well. One of the main limitations of CNNs is that the feature extraction and manipulation of a matrix is necessarily local and inputs have a fixed size - the network cannot handle different sized matrices as input. We need an input data structure that has connections without the constraint of locality within a grid and where you can have an arbitrary number of input features. This can be achieved nicely with the utilization of graphs.

3.2.1 Graphs

We begin by giving the definition of a graph mathematically, seen below:
DEF: A **undirected graph** G is a set of unordered **vertices** v_i and **edges** $e_{i,j}$. The edges are simply ordered pairs of vertices and represent connections between 2 vertices without a specified direction. Vertices and edges are also commonly referred to as **nodes** and **links**.

$$G = \{\mathcal{V}, \mathcal{E}\} \tag{3.20}$$

where $\mathcal{V} = \{v_i\}$ $\mathcal{E} = \{e_{i,j}\}$ $e_{i,j} = \{v_i, v_j\}$

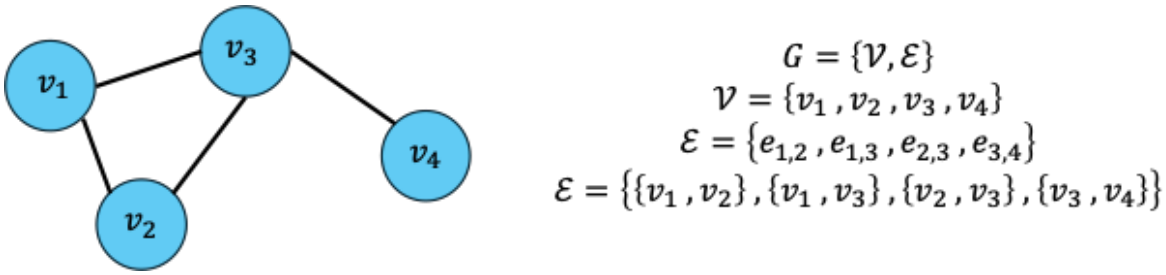


Figure 3.3: Example of a visual graph structure and its mathematical set representation.

We see that a graph is not dependent on vertex or edge ordering within the defined sets, and that the physical orientation of the vertices doesn't matter - the connections between them don't change. This is quite useful for our purposes of modeling atomic systems because if we represent molecules as graphs, these inputs are permutation invariant. We also can input different graphs with differing numbers of vertices and edges altogether, making this structure very robust for handling atomic systems of different sizes.

A CNN is essentially a set of "convolution layers" in which an input matrix is multiplied by by a "convolutional filter" or "kernel" matrix in a specific way to extract structural information that can then be used to construct "better" feature vectors which are then input into a standard FFNN. A **Graph Neural Network (GNN)**, on the other hand, can be thought of as building a graph structure around a FFNN. In order to define a GNN, we must first define a few key terms.

3.2.2 Feature Embedding

In order to utilize graphs for our neural network constructions, we must first define a **feature embedding** for the graph which represents our system. This is essentially a transformation of our raw input data into features which are attached to the various elements of the graph which represents our system. For instance, since we are considering graph networks where the vertices represent atoms, the feature embedding may take an atom's raw data, such as positions, charge, mass, etc., and associate this information with the vertex which represents it in the graph. Note that similar to FFNNs, these features should not be just the raw input data itself in a vector, but intelligently selected functions which serve as a better building block for the network to utilize to construct the learned function we are interested in. We see the general definitions below:

$\mathbf{x}_i^v$ is the **vertex feature vector** of vertex v_i of the input graph

$\mathbf{x}_{i,j}^e$ is the **edge feature vector** of edge $e_{i,j}$ of the input graph

$\mathbf{x}^G$ is the **global feature vector** of the input graph

3.2.3 Message Passing and Pooling Layers

With these definitions, we can now define the central component of a typical GNN, the message passing layer, seen below:

DEF: A **Message Passing Layer** is a layer that generates a new graph representation with updated vertex, edge, and global feature vectors. Each vertex/edge is updated based on the **messages** it receives from the neighboring vertices/edges. The ordering of updates is an architectural choice, but often times the ordering is as follows:

update all edge features $\rightarrow$ update all vertex features $\rightarrow$ update global features

DEF: An **edge message** to a given edge $e_{i,j}$ is simply a function of the 2 vertices that the edge is connecting together - v_i and v_j - and can be defined as follows:

$$m^e_{i,j} = \psi_e(\mathbf{x}^v_i, \mathbf{x}^v_j) \tag{3.21}$$

where ψ_e is the **edge message function**, and is an architectural choice.

DEF: An **edge update** is the procedure for updating the feature vector of an edge based on the edge message it receives from its connected vertices, it's current features, and the graph's global features. It can be defined as follows:

$$\mathbf{x}^e_{i,j}(t+1) = \phi_e(\mathbf{x}^e_{i,j}(t), \mathbf{x}^G(t), m^e_{i,j}(t)) \tag{3.22}$$

where t indicates the layer index in the network and ϕ_e is the **edge update function**.

DEF: A **vertex message** to a given vertex v_i is simply a function of the vertex features of the connected vertex v_j and the edge features of the connecting edge $e_{i,j}$, as follows:

$$m^v_{i,j} = \psi_v(\mathbf{x}^v_j, \mathbf{x}^e_{i,j}) \tag{3.23}$$

where ψ_v is the **vertex message function**, and is an architectural choice.

DEF: A **vertex update** is the procedure for updating the feature vector of a vertex based on vertex messages it receives from its connected vertices/edges, it's current features, and the graph's global features. It can be defined as follows:

$$\mathbf{x}^v_i(t+1) = \phi_v(\mathbf{x}^v_i(t), \mathbf{x}^G(t), \bigoplus_{j \in \mathcal{N}(i)} m^v_{i,j}(t)) \tag{3.24}$$

where t indicates the layer index in the network, ϕ_v is the **vertex update function**, and $\bigoplus$ is a **permutation invariant aggregation operation** - which is a fancy

way of saying an operation that doesn't depend on the ordering of it's arguments. Examples of such operations include summation, averaging, and returning the min or max element of a set. The aggregation of messages is done over the messages from edges/vertices in the **neighborhood** of v_i - meaning the ones connected to v_i by edges.

DEF: **Global messages** to the entire graph are simply functions of the edge and vertex features respectively.

$$M_{i,j}^e = \Psi_e(\mathbf{x}_{i,j}^e) \tag{3.25}$$

$$M_i^v = \Psi_v(\mathbf{x}_i^v) \tag{3.26}$$

where Ψ_e and Ψ_v are the **global message functions**.

DEF: A **global update** is the procedure for updating the global feature vector for the entire graph based on the global vertex and edge messages it receives from all of the vertices and edges in the graph.

$$\mathbf{x}^G(t+1) = \phi_G(\mathbf{x}^G(t), \bigoplus_{e_{i,j} \in G} M_{i,j}^e(t), \bigoplus_{v_i \in G} M_i^v(t)) \tag{3.27}$$

where t indicates the layer index in the network, ϕ_G is the **global update function**, and $\bigoplus$ again is a permutation invariant aggregation operation.

The second type of layer typically seen in GNNs is the pooling layer, defined below:

DEF: A **Graph Pooling Layer** is a layer that reduces the complexity of the graph by breaking the graph into **subgraphs** - local graph structures that collectively compose the larger graph structure - and generating a new, simpler graph representation where these subgraphs are represented as single vertices.

This is analogous to the pooling operations that occur in CNNs, where the image is downsampled to reduce the scale of the data and simplify the structure. Oftentimes in GNNs, the final graph layer is a **Global Pooling Layer** - also called a **readout layer** - where the entire graph is reduced to a new graph representation that just contains a single vertex. See Figure 3.4 for a visualization of the graph pooling operation.

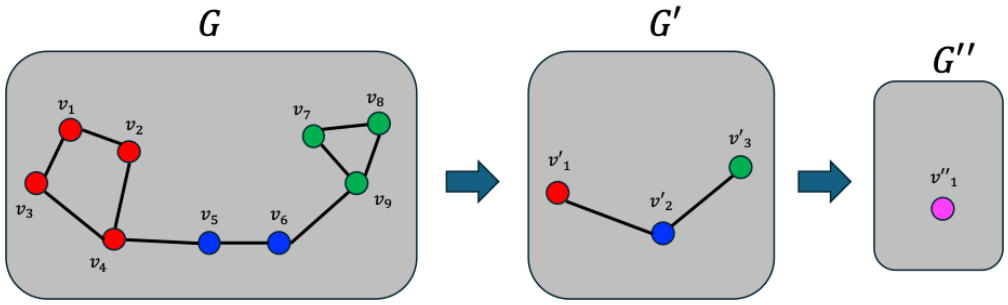


Figure 3.4: A visual example of graph pooling operations, where the arrows represent pooling layers. After the first pooling layer, we see that a colored subgraph of the original graph G is now being represented by a single vertex of the same color in the new output graph G'. The second pooling layer is a global pooling layer, whose output graph G'' is now simply a single vertex that represents the input graph in its entirety.

3.2.4 Message Passing Graph Neural Network

Finally, we can define a general **Message Passing Graph Neural Network (MP-GNN)** as a composition of k message passing and graph pooling layers as follows:

$$\Pi(G) = L^k(L^{k-1}(L^{k-2}(...(L^2(L^1(G)))))) \tag{3.28}$$

where the $L^{[q]}$'s are architecturally defined message passing or pooling layers. In this equation, G is the input graph embedding, i.e. the set of initial $\mathbf{x}_i^v$, $\mathbf{x}_{i,j}^e$, and $\mathbf{x}^G$ that represent the graph, and the graph layers take the input and generate new representations which are then fed into the proceeding layers.

The level of abstraction in this section is impressive, and one may be left wondering where the neural network component to all of this is. If we think of the simplest

case, where we only have vertex features, we see that as we propagate our graph embedding through message passing layers, our vertex features will be "updated" to contain information about not only itself but about its neighboring vertices. If we pass through a pooling layer, we are reducing the complexity of the graph and generating local representations for subgraphs in the network. We can design this architecture for many different purposes, and in the end we will have some final graph feature representation for our specific task. Yet we have not seen how the more basic FFNN components come into play, and how a predictive function is defined and trained using this complex machinery. One key component is that oftentimes the update functions are in fact FFNNs - the "updated" graph feature representations are **learned**. It is also oftentimes the case that the global pooling - or readout - layer at the end is a FFNN as well. The examples in the next section will help clarify some of these abstract concepts.

3.3 Simple Examples

In this section we present two simple examples: a FFNN and a MP-GNN, to help familiarize the reader with these somewhat esoteric concepts.

3.3.1 Feed Forward Neural Network

We can imagine desiring to construct a FFNN architecture which learns a potential energy function for a pair of discrete charges, each with charge q_i and position $\overrightarrow{\mathbf{r}_i}$, where $r_{1,2} = |\overrightarrow{\mathbf{r}_1} - \overrightarrow{\mathbf{r}_2}|$. As mentioned before, it would be a poor choice to simply construct our feature vector to be $\mathbf{x} = [q_1, x_1, y_1, z_1, q_2, x_2, y_2, z_2]^\top$, so instead let's define our feature vector as follows:

$$\mathbf{x}^{[0]} = \begin{bmatrix} q_1 q_2 \\ r_{1,2} \end{bmatrix} \tag{3.29}$$

Now let's define our simple FFNN architecture. Here we note that the superscript k in brackets for the vector $\mathbf{x}$ denotes the input to layer $k+1$, i.e. $\mathbf{x}^{[0]}$ is the feature vector input to the FFNN, $\mathbf{x}^{[1]}$ is the output from layer 1 which is then input to layer 2, and so on.

$$MLP_{\text{charge}}(\mathbf{x}^{[0]}) = \Gamma^{[3]}(\Gamma^{[2]}(\Gamma^{[1]}(\mathbf{x}^{[0]}))) \tag{3.30}$$

$$\Gamma^{[1]}(\mathbf{x}^{[0]}) = \mathbf{x}^{[1]} = \sigma_1(\mathbf{W}^{[1]}\mathbf{x}^{[0]} + \mathbf{B}^{[1]}) \tag{3.31}$$

$$\Gamma^{[2]}(\mathbf{x}^{[1]}) = \sigma_2(\mathbf{W}^{[2]}\mathbf{x}^{[1]} + \mathbf{B}^{[2]}) \tag{3.32}$$

$$\Gamma^{[3]}(\mathbf{x}^{[2]}) = \sigma_3(\mathbf{W}^{[3]}\mathbf{x}^{[2]} + \mathbf{B}^{[3]}) \tag{3.33}$$

We need to define our activation functions for each layer. Our first two layers will have a SiLU activation function, and our final layer will be a linear layer, i.e. with no activation function.

$$\Gamma^{[1]}(\mathbf{x}^{[0]}) = \text{SiLU}(\mathbf{W}^{[1]}\mathbf{x}^{[0]} + \mathbf{B}^{[1]}) \tag{3.34}$$

$$\Gamma^{[2]}(\mathbf{x}^{[1]}) = \text{SiLU}(\mathbf{W}^{[2]}\mathbf{x}^{[1]} + \mathbf{B}^{[2]}) \tag{3.35}$$

$$\Gamma^{[3]}(\mathbf{x}^{[2]}) = \mathbf{W}^{[3]}\mathbf{x}^{[2]} + \mathbf{B}^{[3]} \tag{3.36}$$

We can now write the full FFNN directly as follows:

$$MLP_{\text{charge}}(\mathbf{x}^{[0]}) = \mathbf{W}^{[3]}\left[\text{SiLU}(\mathbf{W}^{[2]}\left[\text{SiLU}(\mathbf{W}^{[1]}\mathbf{x}^{[0]} + \mathbf{B}^{[1]})\right] + \mathbf{B}^{[2]})\right] + \mathbf{B}^{[3]} \tag{3.37}$$

While we now have a full equation, it is still not well defined. We still need to define the dimensions of our three layers. Typically one needs many neurons in the first layer to give the network enough flexibility to "learn" a good approximation to an analytic function. Here we pick 1024 neurons for the first two layers and 1 neuron for the output layer. Now we can complete the definition for our FFNN using summation notation.

$$MLP_{\text{charge}}(\mathbf{x}^{[0]}) = \sum_{i=1}^{1024} W_i^{[3]} \text{SiLU}(\sum_{j=1}^{1024} W_{ij}^{[2]} \text{SiLU}(\sum_{k=0}^{2} W_{jk}^{[1]} x_k^{[0]})) \tag{3.38}$$

Note in the summation notation we have incorporated the bias terms into the sum over the indices of the weight matrix and defined $\mathbf{x}_0^{[k]} = 1$, i.e.

$$\sum_{k=0}^{2} W_{jk}^{[1]} x_k^{[0]} = W_{j0}^{[1]} + W_{j1}^{[1]} x_1^{[0]} + W_{j2}^{[1]} x_2^{[0]} \tag{3.39}$$

where $W_{j0}^{[1]} = B_j^{[1]}$. In this simple construction, we have $W^{[1]} \in \mathbb{R}^{1024\times 2}$, $B^{[1]} \in \mathbb{R}^{1024\times 1}$, $W^{[2]} \in \mathbb{R}^{1024\times 1024}$, $B^{[2]} \in \mathbb{R}^{1024\times 1}$, $W^{[3]} \in \mathbb{R}^{1\times 1024}$, and $B^{[3]} \in \mathbb{R}^{1\times 1}$, for a total of $1,053,697$ weights/biases that will be optimized during training.

For a simple training example with 2048 datapoints generated randomly with noise, we can train our network in an attempt to learn the functional form of the potential energy for two like charges separated by some distance r where $q_1 = q_2 = 1.0$. In Figure 3.5 we see the results of this training example.

In this example, the training data included charge distances between 1.0 and 4.0, and we see that in this range the learned function does a reasonably good job predicting the correct energy. If this were a more serious development project, one could imagine improving this network by designing intelligent features such as polynomial basis functions rather than just using raw data including the charges and their distance. Many other improvements could be made such as using a better optimizer, using LR decay, etc. The Python code for this simple example is provided in Appendix A.1.

3.3.2 Message Passing Graph Neural Network

Now we look at a simple MP GNN model that improves upon the previous example. Let's start by defining the graph embedding of our features. In this example we

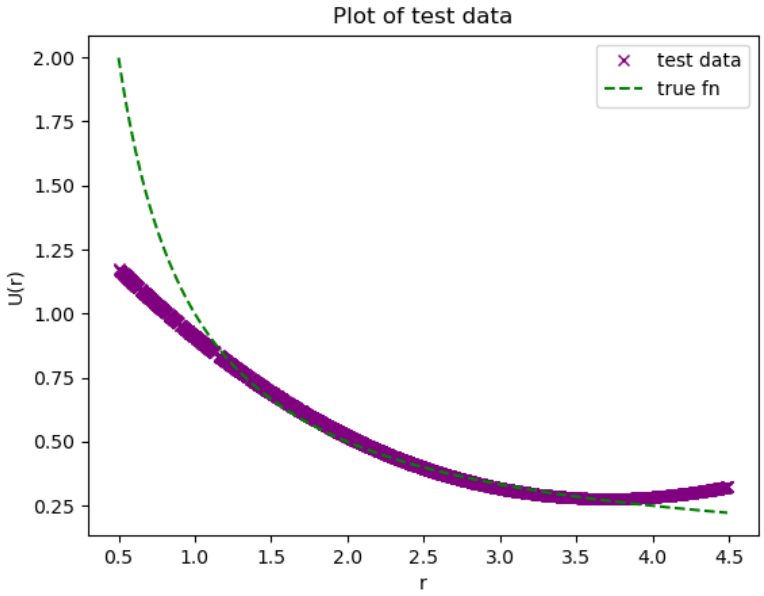


Figure 3.5: Simple FFNN training example demonstrating the flexibility of the neural network construction for learning complicated real valued functional forms from training datasets. Here the data represents the distance between two like charges and their potential energy.

will only have vertex features to make the definitions simpler. We also may wish to add more thought into constructing our features than was given in the previous example. Let's use a set of simple polynomial basis functions $\{P^i(\overrightarrow{\mathbf{r}_i})\}$ as our feature embedding, and use a fixed system of three charges. We can then define our graph as seen below:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\} = \{\{v_1, v_2, v_3\}, \{e_{1,2}, e_{1,3}, e_{2,3}\}\} \tag{3.40}$$

For simplicity, we define the edges to be undirected and we have fully connected our verticies. In this graph definition, our verticies will represent atoms, and the connections will represent interactions. Remember that we are only defining vertex features, and are not considering edge or global features. These can be defined as seen below, where we explicitly choose to use only three polynomial basis functions:

$$\mathbf{x}_i^v(0) = \begin{bmatrix} q_i \\ P^1(\overrightarrow{\mathbf{r}_i}) \\ P^2(\overrightarrow{\mathbf{r}_i}) \\ P^3(\overrightarrow{\mathbf{r}_i}) \end{bmatrix} \tag{3.41}$$

In this example we have taken our raw inputs and constructed features with which to better represent our system - in this case, using polynomial functions of the atomic position. We can see how this construction can greatly improve the ability of the network to learn analytic functional forms. For each atom i, we have now defined a feature embedding for that atom.

Next we look at how we want to define our message passing operation with which we update our graph features. Since we only have vertex features in this example, we define a vertex message from atom j to atom i as a function of only the vertex feature of atom j, $\mathbf{x}_j^v$. The vertex message $m_{i,j}^v$ is defined below:

$$m_{i,j}^v = \psi_v(\mathbf{x}_j^v) = \mathbf{x}_j^v \tag{3.42}$$

For simplicity, we define the message to be the vertex feature vector itself. We then can define a simple vertex update as follows:

$$\mathbf{x}_i^v(t+1) = \phi_v \left(\mathbf{x}_i^v(t) || \frac{1}{N_i} \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j^v(t) \right) \tag{3.43}$$

Here we have introduced $||$ which is concatenation, and N_i as the number of atoms in the neighborhod of i. We see our permutation invariant aggregation operation is simply taking the average of the messages from neighbors j. We define ϕ_v to be a standard FFNN with - for ease of notation - 1 layer with SiLU activation function and 4 neurons, which is connected to one linear output layer. This is given below:

$$\phi_v(\mathbf{z}_i^v(t)) = \mathrm{MLP}_{\mathrm{mp}}(\mathbf{z}_i^v(t)) = \mathbf{W}_{\mathrm{mp}}^{[2]}\left[\mathrm{SiLU}(\mathbf{W}_{\mathrm{mp}}^{[1]}\mathbf{z}_i^v(t) + \mathbf{B}_{\mathrm{mp}}^{[1]})\right] + \mathbf{B}_{\mathrm{mp}}^{[2]} \tag{3.44}$$

Here for ease of notation we have defined the vector $\mathbf{z}_i^v(t)$ to be the following:

$$\mathbf{z}_i^v(t) = \mathbf{x}_i^v(t)||\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j^v(t) \tag{3.45}$$

Now we have a very simple and well defined message passing operation which will combine vertex features between connected verticies in our message passing layer. Let's assume we have two message passing layers and a readout layer which is itself a FFNN, which we will denote E. We can finally define our MP-GNN as follows:

$$\Pi(G) = E(L^{[2]}(L^{[1]}(G))) \tag{3.46}$$

where G is the initial graph embedding, i.e.:

$$G = \begin{bmatrix} \mathbf{x}_1^v(0) \\ \mathbf{x}_2^v(0) \\ \mathbf{x}_3^v(0) \end{bmatrix} \tag{3.47}$$

and where we define E as:

$$E = \sum_i E_i = \sum_i \mathrm{MLP}_{\mathrm{out}}(\mathbf{x}_i^v(2)) = \sum_i \mathbf{W}_{\mathrm{out}}^{[2]}\left[\mathrm{SiLU}(\mathbf{W}_{\mathrm{out}}^{[1]}\mathbf{z}_i^v(2) + \mathbf{B}_{\mathrm{out}}^{[1]})\right] + \mathbf{B}_{\mathrm{out}}^{[2]} \tag{3.48}$$

In going through this simple example, we see how quickly the construction becomes increasingly complex. We note that there is still ambiguity in our architectural construction: the vertex update function ϕ_v and the readout layer E, both of which have been constructed as FFNNs, have undefined dimensions. The SiLU layer could have

10, 20, or 1000 neurons. The linear layer could output a single number, 10 numbers, or 100 numbers. These are intentionally left ambiguous in this example to allow for flexibility and user fine tuning. It is implied that the various dimensions must match up - we can't feed 10 numbers that are output from function A into another function B which takes 100 numbers as input, after all.

This example has no corresponding Python code or demonstration, but we take a moment to describe the preceding definitions. This MP-GNN can be visualized with the computational flowchart below in Figure 3.6.

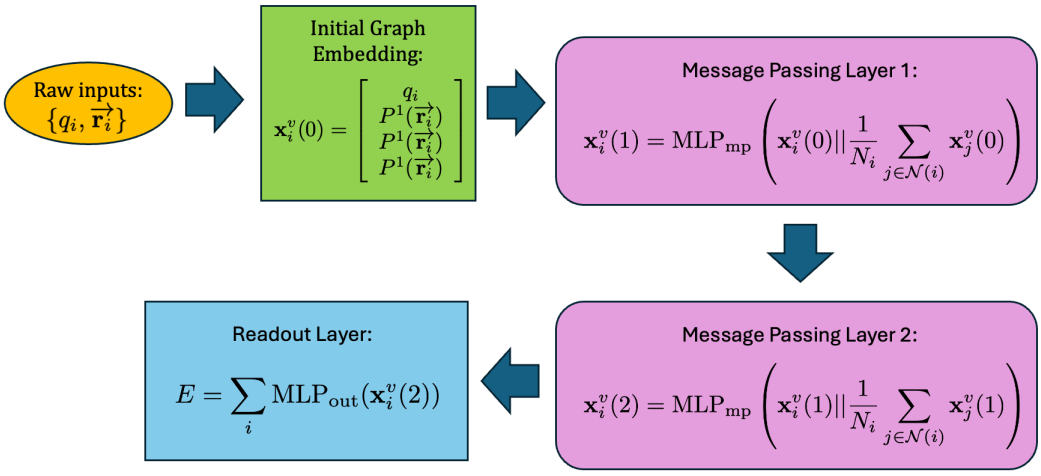


Figure 3.6: Computational flowchart of the simple architecture we have defined to demonstrate the functionality of MP-GNNs.

Here we see that we first take our raw input data - in this case for each particle we have a charge and a position - and then we embed this information into our graph structure to define the initial vertex feature vectors $\mathbf{x}_i^v(0)$ for each charge. These are composed of functions of this raw input data, which in this case are polynomial basis functions. These initial vertex feature vectors are then updated through two message

passing layers which contain FFNNs, resulting in a final learned feature vector $\mathbf{x}_i^v(2)$ for each charge. These learned feature vectors are then fed into the readout layer, where they are used to learn an atomic energy for each atom, the sum of which defines the total energy of the charge configuration.

We see that through this construction, not only are we extending the FFNN construction by learning the features which go into the network, but we are imposing physically meaningful structure onto them. This architecture was given as an example with a concrete number of atoms, but by definition, it is not restricted to this specific number of atoms, and is also permutation invariant - the ordering of the vertices and edges within the graph set does not matter. Though simple, this example illustrates the power of graph networks for greatly improving the capabilities of neural networks and imposing physically meaningful structure to the learning process.

One limitation that must now be mentioned regarding GNNs, is that although the input graphs which represent atomic systems are not size constrained and are permutation invariant, the predictive function defined by the GNN may be dependent on the input system's spatial orientation. For instance, in the simple GNN example here, the vertex features are simple polynomials. If you imagine rotating the system such that a particular atom now has position $\mathbf{r}_i'$ instead of the original position $\mathbf{r}_i$, it is clear that its initial embedding and by extension, its predicted energy, could change. In principle, given enough examples, the network should "learn" that the energy of a particular atom is dependent only on the relative positions of the neighboring atoms, but this is not baked into the formalism in general. In the next section we address this issue by introducing the concepts of invariance and equivariance in the context of GNNs.

Chapter 4: Neural Networks in Physics

Now that we have defined the basic formalism behind Neural Networks, we can begin to look at how they are actually used to build MLIPs. In this chapter, we will go over some important design considerations borrowed from the mathematical field of **abstract algebra** - mainly **group theory** - which are used to build MLIP architectures. We will see how several MLIPs have been constructed utilizing these concepts in an early work by Behler and Parrinello [77], as well as in newer architectures including Nequip [78] and Allegro [79].

4.1 Group Theory

When constructing a neural network as a machine learned interatomic potential, we want to ensure that the architecture respects the symmetry of the system. This learned potential should be constructed such that it gives the same prediction whether or not you rotate, translate, or invert the input atomic coordinates. More formally, we can say that scalar properties of a system, such as the system's total energy, are **invariant** with respect to the group of continuous rotations/translations/reflections/inversions in 3D Cartesian space, the **Euclidean Group (3)** $E(3)$. We also note that vector properties of a system are **equivariant** with respect to the group $E(3)$ - meaning they change along with the rotation, translation, reflection, or inversion. We can define functions that are invariant or equivariant with respect to these rotations and inversions more formally with **group theory**, which we describe here in part based on Tinhkam's "Group Theory and Quantum Mechanics" textbook [80], Dummit and Foote's "Abstract Algebra" textbook [81], Smith's "Introduction to Abstract Algebra" [82], and Rotman's "An Introduction to the theory of Groups" [83]. Here

we borrow notation from mathematics, where $\forall$ means "for all", $\in$ means "in" , $\exists$ means "exists", and $|$ means "such that". By doing so we can see how equivariant neural networks which respect the symmetry of the atomic system can be constructed to better represent its physical properties.

4.1.1 Definition of a Group

DEF: A **group** $\mathcal{G} = (G, \star)$ is simply an abstract set of elements G coupled with an operation $\star$ with the following properties:

$$\forall a, b \in G,\ a \star b \in G \tag{4.1}$$

$$\exists e \in G \mid \forall a \in G,\ e \star a = a \star e = a \tag{4.2}$$

$$\forall a \in G,\ \exists a^{-1} \in G \mid a \star a^{-1} = a^{-1} \star a = e \tag{4.3}$$

$$\forall a, b, c \in G,\ (a \star b) \star c = a \star (b \star c) \tag{4.4}$$

These properties define closure of the group under the operation $\star$, the existence of an identity e under the $\star$ operation, the existence of an inverse for every element in the group under the $\star$ operation, and the restriction that $\star$ must be associative. Note that $\star$ need not be commutative, which is indeed the case for the group we are interested in, **E(3)**.

4.1.2 Group Representation Theory

DEF: A **group action on a vector space** V defines a **representation of a group** $\mathcal{G}$, is denoted by ϕ_V and defined as follows:

$$\phi_V : G \times V \to V \tag{4.5}$$

where $\times$ is the **cartesian product** of the sets G and V, which is defined as the set $\{(g_i, v_j) \mid g_i \in G, v_j \in V\}$. This means that for every group element $g \in G$ and every vector $v \in V$, there is a mapping defined as follows:

$$\phi_V(g, v) = v' \in V \tag{4.6}$$

where $\phi_V(g, v)$ performs the group action of g on v and outputs v'.

A simple example can help illustrate these abstract concepts. Here we look at the rotational group of an equilateral triangle, C_3. Let $\mathcal{G} = (G, \circ)$ where $G = \{e, a, b\}$ and $\circ$ is the composition operation. Here we define e as the identity element, a as a 120 degree rotation, and b as a 240 degree rotation. The rotations are in a fixed $2D$ plane around a common axis of rotation. We can define a group representation as follows, given $g \in G$ and $\mathbf{x} \in \mathbb{R}^2$:

$$\phi_{\mathbb{R}^2} : G \times \mathbb{R}^2 \to \mathbb{R}^2 \mid \phi(g, \mathbf{x}) \equiv D_g \mathbf{x} \tag{4.7}$$

where D_g is a **2D rotation matrix** defined below:

$$D_e = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad D_a = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \qquad D_b = \begin{bmatrix} -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \tag{4.8}$$

We note that $b = a^{-1}$ and the group properties all hold. While this is one representation of the group, there are potentially several others that can be determined and analyzed using various theorems presented in a more complete presentation of the subject, which is outside of the scope of this dissertation. We can, for instance, find a 1-dimensional representation in $\mathbb{C}$ as follows, given $x \in \mathbb{C}$:

$$\phi_{\mathbb{C}} : G \times \mathbb{C} \to \mathbb{C} \mid \phi(g, x) \equiv c_g x \tag{4.9}$$

where c_g is defined below:

$$c_e = 1 \qquad c_a = \omega \qquad c_b = \omega^2 \qquad \omega = e^{\frac{2\pi i}{3}} \tag{4.10}$$

If you apply the group action of a to a vector $x \in \mathbb{C}$ by multiplying it by c_a, we see that it rotates this complex number by $2\pi/3$ radians - exactly 120 degrees - as expected.

We also note that representations can be either **reducible** or **irreducible**, where an irreducible representation is defined as one that cannot be broken down into a direct sum of representations of smaller dimension through application of a unitary transformation to all members of the representation.

4.1.3 Invariance and Equivariance

Now that we have defined groups and group representations in the most general way, we can look at how invariance and equivariance are formally defined.

DEF: A function $f : V \to W$ is **invariant** with respect to a group action if the function gives the same result whether or not a group action is applied, i.e.:

$$\forall g \in G,\ v \in V,\ f(\phi_V(g, v)) = f(v) \tag{4.11}$$

DEF: A function $f : V \to W$ is **equivariant** with respect to a group action if the function's output transforms the same way as the input, i.e. it doesn't matter the order of the group action and function application:

$$\forall g \in G,\ v \in V,\ \phi_W(g, f(v)) = f(\phi_V(g, v)) \tag{4.12}$$

For clarification on these concepts, let's look at a simple example regarding invariance.

Let's consider the aforementioned group C^3 and the group action $\phi_{\mathbb{R}^2}$, along with the distance function in $\mathbb{R}^2$. If we define $\mathbf{r} \in \mathbb{R}^2$ using the typical notation, i.e. $\mathbf{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}}$, then the distance function is defined as follows:

$$d(\mathbf{r}) \equiv \sqrt{x^2 + y^2} \tag{4.13}$$

We can see that if we apply a group action, i.e. a 2-D rotation by 120 degrees, before applying our function, we get the following:

$$d(\phi_{\mathbb{R}^2}(g, \mathbf{r})) = d(D_a\mathbf{r})) = d\left(\begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}\right) = d\left(\begin{bmatrix} -\frac{1}{2}x - \frac{\sqrt{3}}{2}y \\ \frac{\sqrt{3}}{2}x - \frac{1}{2}y \end{bmatrix}\right) \tag{4.14}$$

$$d(\phi_{\mathbb{R}^2}(g, \mathbf{r})) = \sqrt{\left(-\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right)^2 + \left(\frac{\sqrt{3}}{2}x - \frac{1}{2}y\right)^2} = \sqrt{\left[\left(\frac{1}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2\right](x^2 + y^2)} \tag{4.15}$$

$$d(\phi_{\mathbb{R}^2}(g, \mathbf{r})) = \sqrt{[1]\,(x^2 + y^2)} = \sqrt{x^2 + y^2} = d(\mathbf{r}) \tag{4.16}$$

Thus, the criteria for invariance is met under the 120 degree rotation. Since successive 120 rotations are equivalent to 240 and 360 degree rotations, the criteria is also met for the other group elements, and the distance function is invariant under the group action. One could imagine a similar check to see if a function is equivariant, though we will not present an example for this case here.

4.1.4 Continuous Groups and E(3)

The previous discussion on groups has implicitly assumed that groups have a finite number of elements. The group $E(3)$ is different in that it is a **continuous group** rather than a **finite group**. Continuous groups, often called **Lie groups**, are extensions of the ideas presented above for finite groups, but for groups containing infinite

sets. While a full treatment of this topic is beyond the scope of this dissertation, to get a feel for the concept we can show an example of a Lie group we all know quite well - the set of real numbers $\mathbb{R}$ coupled with the standard addition operator $+$:

$$\forall a, b \in \mathbb{R},\ a + b \in \mathbb{R} \tag{4.17}$$

$$\exists e \in \mathbb{R} \mid \forall a \in \mathbb{R},\ e + a = a + e = a \tag{4.18}$$

$$\forall a \in \mathbb{R},\ \exists a^{-1} \in \mathbb{R} \mid a + a^{-1} = a^{-1} + a = e \tag{4.19}$$

$$\forall a, b, c \in \mathbb{R},\ (a + b) + c = a + (b + c) \tag{4.20}$$

Clearly, the set of real numbers meet the requirements to be a group as defined above, where e is the additive identity 0. For the group $E(3)$, the group elements turn out to be continuous generators (functions) of rotations, translations, reflections, and inversions. The subgroup of $E(3)$, $SO(3)$, is the group of continuous rotations about a point. It turns out the spherical harmonics Y_ℓ^m are basis functions for irreducible representations of $SO(3)$.

4.1.5 Tensor Products

Let's imagine we want to construct a message passing graph network with learned features that are equivariant with respect to the rotation group $SO(3)$. If we choose these features to be irreducible representations of $SO(3)$, and then intelligently combine them during our message passing operation, we ensure that our neural network is equivariant with respect to $SO(3)$. This special way of combining them involves

utilizing the tensor product of representations. To understand this, we start with the more general notion of a tensor product.

The **tensor product** of two vector spaces V and W is the vector space denoted by $V \otimes W$, where there is a **multi-linear map** defined as follows:

$$\phi : V \times W \to V \otimes W \mid \phi(\mathbf{v}, \mathbf{w}) = \mathbf{v} \otimes \mathbf{w} \tag{4.21}$$

Here we are essentially saying that for each vector $\mathbf{v} \in V$ and $\mathbf{w} \in W$, ϕ maps these uniquely to a vector denoted $\mathbf{v} \otimes \mathbf{w}$ which is an element of the vector space $V \otimes W$. A general element of $V \otimes W$ is called a **tensor**, and $\mathbf{v} \otimes \mathbf{w}$ is called the **tensor product** of $\mathbf{v}$ and $\mathbf{w}$.

To understand this more concretely, we can define the tensor product of two vectors in terms of their chosen basis sets. Assume that we have two vectors $\mathbf{v} \in V$ and $\mathbf{w} \in W$, with chosen basis sets $B_V = \{\mathbf{b}_i^V\}$ and $B_W = \{\mathbf{b}_j^W\}$. These means we can express these two abstract vectors as follows:

$$\mathbf{v} = \sum_i v_i \mathbf{b}_i^V = v_1 \mathbf{b}_1^V + v_2 \mathbf{b}_2^V + ... \tag{4.22}$$

$$\mathbf{w} = \sum_j w_j \mathbf{b}_j^W = w_1 \mathbf{b}_1^W + w_2 \mathbf{b}_2^W + ... \tag{4.23}$$

If we then take the tensor product $\mathbf{v} \otimes \mathbf{w}$, we can express it as follows:

$$\mathbf{v} \otimes \mathbf{w} = \sum_i \sum_j v_i w_j \quad \mathbf{b}_i^V \otimes \mathbf{b}_j^W \tag{4.24}$$

What's nice about this definition is that it allows us to naturally form a basis for the vector space $V \otimes W$ as follows:

$$B_{V\otimes W} = \{\mathbf{b}_i^V \otimes \mathbf{b}_j^W\} = \{\mathbf{b}_1^V \otimes \mathbf{b}_1^W, \mathbf{b}_2^V \otimes \mathbf{b}_1^W, ..., \mathbf{b}_1^V \otimes \mathbf{b}_2^W, \mathbf{b}_2^V \otimes \mathbf{b}_2^W, ..., \mathbf{b}_1^V \otimes \mathbf{b}_3^W, \mathbf{b}_2^V \otimes \mathbf{b}_3^W, ...\} \tag{4.25}$$

To relate this abstract concept to objects we are familiar with in physics, we note that we can define two position vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ in terms of their basis vectors $B_{\mathbb{R}^3} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ as follows:

$$\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3 \tag{4.26}$$

$$\mathbf{b} = b_1\mathbf{e}_1 + b_2\mathbf{e}_2 + b_3\mathbf{e}_3 \tag{4.27}$$

where we use the notation $\mathbf{e}_1 = \hat{\mathbf{x}}$, $\mathbf{e}_2 = \hat{\mathbf{y}}$, and $\mathbf{e}_3 = \hat{\mathbf{z}}$ for convenience. We can then define the tensor product of $\mathbf{a}$ and $\mathbf{b}$ as follows:

$$\mathbf{a} \otimes \mathbf{b} = \sum_i \sum_j a_i b_j \quad \mathbf{e}_i \otimes \mathbf{e}_j \tag{4.28}$$

We again note that these vectors denoted $\mathbf{e}_i \otimes \mathbf{e}_j$ naturally form a set of basis vectors with which we can represent our tensor product as an element of the vector space $\mathbb{R}^3 \otimes \mathbb{R}^3$, which is shown below:

$$\mathbf{a}\otimes\mathbf{b} = a_1b_1\mathbf{e}_{11}+a_2b_1\mathbf{e}_{21}+a_3b_1\mathbf{e}_{31}+a_1b_2\mathbf{e}_{12}+a_2b_2\mathbf{e}_{22}+a_3b_2\mathbf{e}_{32}+a_1b_3\mathbf{e}_{13}+a_2b_3\mathbf{e}_{23}+a_3b_3\mathbf{e}_{33} \tag{4.29}$$

where we have changed notation for convenience by defining $\mathbf{e}_{ij} = \mathbf{e}_i \otimes \mathbf{e}_j$. We note that this looks suspiciously like a matrix, and in fact, $\mathbb{R}^3 \otimes \mathbb{R}^3$ turns out to be isomorphic to the vector space of 3×3 matrices, $\mathbb{R}^{3\times3}$.

We can more generally define a tensor T of type (p, q) and rank $p + q$ as the multilinear map itself:

$$T : \bigotimes_{i=1}^{p} V \otimes \bigotimes_{j=1}^{q} W \to Z \tag{4.30}$$

where the larger tensor operator implies iteration of tensor products, i.e. $\bigotimes_{i=1}^{3} V = V \otimes V \otimes V$. Here p is the **contravariance** order and q is the **covariance** order, which determine how the objects behave geometrically under change of basis transformations. A covariant vector will transform in the same way as the basis vectors, whereas a contravariant vector will transform opposite to the change of basis. Though we will not go into further detail at this point, we can see the beauty of this abstract and quite general construction: scalars can be considered type $(0, 0)$ tensors, vectors can be considered type $(1, 0)$ tensors, and linear maps represented by matrices can be considered type $(1, 1)$ tensors. It turns out even inner products can be thought of a type $(0, 2)$ tensor defining a mapping from $\mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$.

We now wish to consider the irreducible representations of $SO(3)$. These can be expressed in terms of basis functions which are the spherical harmonics Y_ℓ^m. We can express an irreducible representation of $SO(3)$ of degree ℓ and dimension $2\ell + 1$ as follows:

$$\mathbf{x} = \sum_m x_m Y_\ell^m \tag{4.31}$$

where m is an integer such that $-\ell \leq m \leq \ell$. Now suppose we have two distinct irreducible representations of different degrees that we wish to combine. We can do this with the tensor product of representations, as defined below:

$$(\mathbf{x} \otimes \mathbf{y})_{\ell_3,m_3} = \sum_{m_1,m_2} \mathcal{W}_{\ell_1,\ell_2,\ell_3}^{m_1,m_2,m_3} x_{\ell_1,m_1} y_{\ell_2,m_2} \tag{4.32}$$

Here we have introduced the Wigner $3-j$ symbol, defined below:

$$\mathcal{W}_{\ell_1,\ell_2,\ell_3}^{m_1,m_2,m_3} = \begin{pmatrix} \ell_1 & \ell_2 & \ell_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \tag{4.33}$$

where we have defined the order of $\mathbf{x}$ to be ℓ_1 and the order of $\mathbf{y}$ to be ℓ_2. We note that m_1 and m_2 are integers such that $-\ell_1 \leq m_1 \leq \ell_1$ and $-\ell_2 \leq m_2 \leq \ell_2$. Furthermore, the product indices ℓ_3 and m_3 must satisfy the inequalities present when adding angular momentum, i.e. $|\ell_1 - \ell_2| \leq \ell_3 \leq \ell_1 + \ell_2$ and $-(m_1 + m_2) \leq m_3 \leq m_1 + m_2$.

4.2 Selected Example Neural Network Architectures

In this section, we review several important neural network architectures that have been successfully implemented to construct MLIPs, showcasing the capabilities of these methodologies to maintain similar accuracy to DFT while greatly improving computational efficiency.

4.2.1 Behler and Parrinello Neural Network

Starting with basic FFNNs, we look at a very nice example of early work done by Behler and Parrinello [77]. Their approach takes atomic features G_i^1 and G_i^2 for each atom in a given system, feed them through a FFNN, and predicts their energies. The total energy of the system is simply the sum of all of the per-atom energies. Importantly, each atomic FFNN is identical - meaning they all have the same number of layers, neurons per layer, and weights and biases. The reason for this restriction is so that the model can be permutation invariant - it shouldn't matter which atom

we label 1 and which atom we label 10. The loss function is the L2 loss, which they minimize through standard gradient descent backpropagation. They use the hyperbolic tangent activation function for all layers except the output layer, which is a linear layer. See Figure 4.1 for a reproduced visualization of their NN architecture.

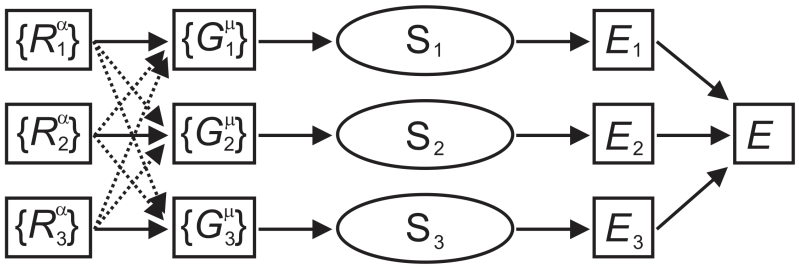


Figure 4.1: Visualization of the FFNN architecture as reproduced from reference [77] with permission.

One of the most important choices in their architecture are the features themselves. As mentioned in the section 3.3.5, it is not ideal to simply use raw inputs - mass, charge, position, etc. - as features, but to instead build functional representations of these inputs as features that will be used as building blocks by the FFNN to build the predictive function. In this architecture, they choose the following feature definitions:

$$G_i^1 = \sum_{j \neq i} e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij}) \tag{4.34}$$

$$G_i^2 = 2^{1-\zeta} \sum_{j,k \neq i} \left[(1 + \lambda cos(\theta_{i,j,k}))^\zeta\right] e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \tag{4.35}$$

$$f_c(R_{ij}) = \begin{cases} \frac{1}{2}\left[cos(\frac{\pi R_{ij}}{R_c} + 1)\right] & if \quad R_{ij} \leq R_c \\ 0 & if \quad R_{ij} > R_c \end{cases} \tag{4.36}$$

The purposes of G^1 and G^2 are to generate features to represent the $i-th$ atom in the system which are then fed into a FFNN and used to predict the energy of

that atom. The features function as an atomic representation of the atom i and all neighbors j - where G^1 represents radial functions and G^2 represents angular functions. G^1 is a sum of gaussian functions which are dependent on the distance between i and j, R_{ij}, are centered at R_s, and are multiplied by a cutoff function f_c. G^2 is the sum of the cosines of the angles between all possible triplets (i, j, k) and is multiplied by a gaussian and 3 cutoff functions f_c. η, R_s, ζ, and λ are all fixed constants that are chosen before training and have nothing to do with the FFNN learning.

During the testing of their FFNN architecture they focused on bulk Si, training on roughly 8000 DFT structures through their FFNN comprised of 2 hidden layers with approximately 40 neurons per layer. For their testing of the FFNN they tried different values of the fixed parameters η, R_s, ζ, and λ in the input features G^1 and G^2. The FFNN training optimized a few thousand weights to build a model that was then tested against DFT and a few emperical potentials. They computed a **Radial Distribution Function (RDF)** for Si and they also looked at the energy errors between their model predictions and DFT. The NN yielded a predicted RDF that was very close to the DFT results - much better than the emperical potentials. Comparing the energies and forces to DFT, they found good agreement, with errors around $4 - 6$ meV per atom for the energies and roughly 0.2 eV per anstrom for the forces. This is one of the first results that really shows that NNs can approach accuracies of DFT calculations while retaining the speed and scalability of emperical potentials. See Figure 4.2 for reproduced plots of their results.

4.2.2 MEGNET Graph Network Architecture

While Behler and others have shown that FFNNs are very promising, advancements have been made utilizing the various extensions to FFNNs, including CNNs and

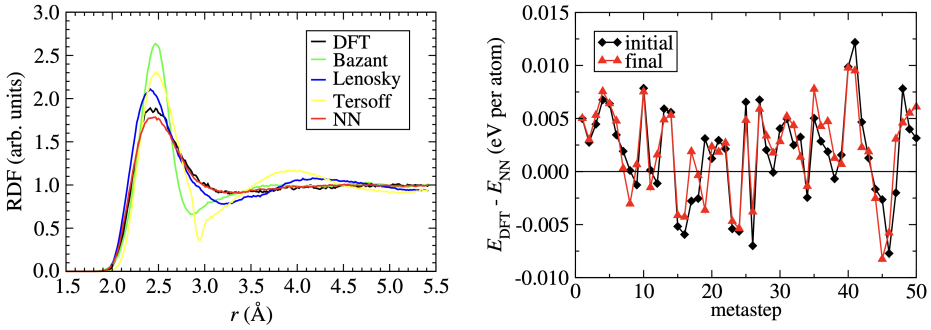


Figure 4.2: Computed RDF functions for various methods (left) and energy per atom errors of the NN approach compared to DFT (right), as reproduced from reference [77] with permission.

more recently, GNNs. Graph networks are the "natural" structure for molecular modeling, and seem to be delivering excellent results. A very nice example of basic GNN implementation is by Chen et al. [26] In this paper, they present their graph network architecture they have named **MatErials Graph NETwork (MEGNET)**. This network is a message passing graph network where the graph vertices represent atoms, the edges represent bonds between atoms, and the graph represents either a molecule or a crystalline structure. The graph embedding includes vertex features, edge features, and global features, which are denoted by v_i, e_k, and u respectively, where k represents a given (i, j) pair. They include a lot of input data, which is listed in Table 1 of their paper, including atomic species, chirality, electronic orbital configuration, bond type, bond length, and several others. A visualization of their architecture is given in Figures 1 and 2 in their publication [26].

They update the graph through messages passing layers by first updating the edge features, then the vertex features, then the global features. They embed their message passing operations in what they call the MEGNET layer, which pre-processes the input features by feeding them into FFNN layers before performing the message passing updates. Once several MEGNET layers have been iterated, they then flatten

the graph embedding via a global pooling layer into a vector that is then fed into a final FFNN, which outputs a predicted material property. The message passing update functions are defined below:

$$e_k^{[l+1]} = \phi_e(e_k^{[l]} \bigoplus u^{[l]} \bigoplus v_{s_k}^{[l]} \bigoplus v_{r_k}^{[l]}) \tag{4.37}$$

$$v_i^{[l+1]} = \phi_v(v_i^{[l]} \bigoplus u^{[l]} \bigoplus \frac{1}{N_i^e} \sum_{k=1}^{N_i^e} e_{k \in \mathcal{N}(i)}^{[l+1]}) \tag{4.38}$$

$$u^{[l+1]} = \phi_u(u^{[l]} \bigoplus \frac{1}{N^e} \sum_{k=1}^{N^e} e_k^{[l+1]} \bigoplus \frac{1}{N^v} \sum_{i=1}^{N^v} v_i^{[l+1]}) \tag{4.39}$$

Here they have defined $\bigoplus$ as **concatenation** (i.e. for $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, $x \bigoplus y = \{x_1, ..., x_n, y_1, ..., y_m\}$), s_k and r_k as the connecting atoms to the edge e_k, N_i^e as the number of edges connected to v_i, and N^e and N^v as the number of edges and vertices in the graph respectively. The update functions ϕ_e, ϕ_v, and ϕ_u are all FFNNs with 2 hidden layers with modified softplus activation functions.

What this complicated looking set of formulas really means is that they are first updating the edge feature vectors to be learnable functions of the original features of the edge itself, the neighboring atoms' features, and the global features. Now that the edges have gained information in this way, the atoms' feature vectors are then updated in a similar way - to be learnable functions of their original features and all of their edges' features, as well as the global features. In this way, they are indirectly "learning" about their neighboring atoms, and after N message passing operations, they will "know" about neighboring atoms that are N hops away. Finally, the global features are updated to be learned functions of all of the newly updated edge and atom features.

In their study, they assess the MEGNET architecture for its ability to predict various material properties, which can be seen in Table 2 in their paper [26], where

they determined that their architecture performs competitively with SchNet and enn-s2s, two other neural network based approaches, and exceeds their target values.

4.3 Adopted Neural Network Architectures

In this section, we introduce and briefly describe the neural network architectures utilized in our research, which were developed by the Materials Intelligence Research group at Harvard University: NequIP and Allegro.

4.3.1 NequIP Architecture

The NequIP architecture [78] is a ground breaking message passing graph neural network architecture which has had great success in maintaining first principles accuracy while drastically decreasing computational time for molecular dynamics simulations. It has had a massive overhaul recently to further improve the efficiency and capability of the software [84]. A flowchart depicting the architecture is reproduced below in Figure 4.3:

NequIP uses atom centered graph embeddings, where atoms are represented by verticies in the graph structure. The associated vertex features, often called "atomic representations", are denoted by $\mathbf{V}_i^{[k]}$, where i is the atom index and k is the "NequIP layer" index. The initial embeddings ($k = 0$) are learned functions of the input atomic numbers Z_i with components as seen below:

$$V_{i,c,\ell=0,m=0}^{[0]} = \mathrm{MLP}_{\mathrm{embed}}(Z_i) \tag{4.40}$$

These initial embeddings are then fed through a series of "interaction blocks" which define the "NequIP layers", where c is the channel index, and ℓ, m are the representation degree and order. The general atomic representations at layer k, $\mathbf{V}_i^{[k]}$, are importantly constructed from irreducible representations of the $O(3)$ symmetry

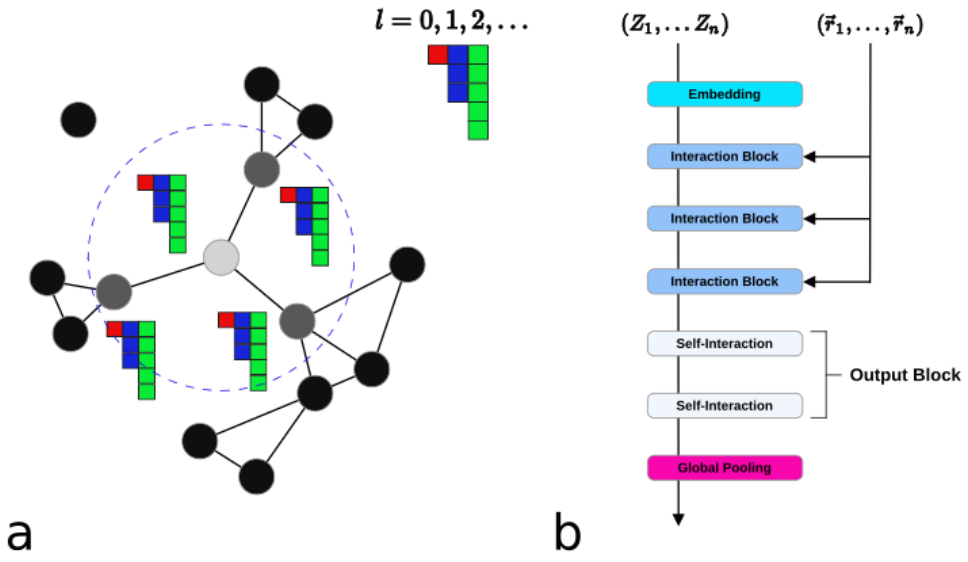


Figure 4.3: Visualization of the NequIP architecture as reproduced from reference [78] with permission.

group. The interaction block intelligently combines these atomic representations of a given atom i with the atomic representations of the neighboring atoms within some fixed cutoff radius r_{cut}, through what is called a convolution - which in this context defines the message passing operation in the network. Contained in this convolution is what is called a convolutional "filter", $S_m^\ell(\overrightarrow{\mathbf{r}}_{i,j})$, which is defined below:

$$S_m^\ell(\overrightarrow{\mathbf{r}}_{a,j}) = R(r_{ij}) Y_m^\ell(\hat{\mathbf{r}}_{ij}) \tag{4.41}$$

where we have introduced the index j which represents a particular neighboring atom, i.e. $j \in \mathcal{N}(i)$. The $Y_m^\ell(\hat{\mathbf{r}}_{ij})$ functions are the familiar spherical harmonics, which serve as the basis functions for the irreducible representation of $O(3)$. The $R(r_{ij})$ functions are learned coefficients of the spherical harmonics, which are defined below:

$$R(r_{ij}) = MLP_{\text{conv}}(B(r_{ij})) = W_n(\sigma(...(\sigma(W_1(\mathbf{B}(r_{ij})))))) \tag{4.42}$$

where the W's are the learned parameters and σ is the activation function, which is typically SiLU. This learned function is itself constructed from a set of invariant radial basis functions $\mathbf{B}(r_{ij}) = \{B_b(r_{ij})\}$, which are Bessel functions of combined with the smooth polynomial envelope function f_{env} defined in the DIMENET architecture [85], as seen below:

$$B_b(r_{ij}) = \frac{2}{r_c} \frac{\sin\left(\frac{b\pi}{r_c}\right)}{r_{ij}} f_{\text{env}}(r_{ij}, r_c; p) \tag{4.43}$$

where r_c is the local cutoff radius and f_{env} is the polynomial defined below, with parameter p typically chosen to be equal to 6:

$$f_{\text{env}}(r_{ij}, r_c; p) = 1 - \frac{(p+1)(p+2)}{2}\left(\frac{r_{ij}}{r_c}\right)^p + p(p+2)\left(\frac{r_{ij}}{r_c}\right)^{p+1} - \frac{p(p+1)}{2}\left(\frac{r_{ij}}{r_c}\right)^{p+2} \tag{4.44}$$

This convolutional filter $S_m^\ell(\overrightarrow{\mathbf{r}}_{a,j})$ is then used to define the full convolution $\mathcal{L}$ at layer k, seen here:

$$\mathcal{L}_{i,c,m_3}^{\ell_3,\ell_2,\ell_1}(\overrightarrow{\mathbf{r}}_i, V_{i,c,\ell_1,m_1}^{[k-1]}) = \sum_{m_1,m_2} C_{\ell_1,m_1,\ell_2,m_2}^{\ell_3,m_3} \sum_{j\in\mathcal{N}(i)} R(r_{ij})_{c,\ell_1,m_1,\ell_2,m_2,\ell_3,m_3} Y_{m_2}^{\ell_2}(\hat{\mathbf{r}}_{ij}) V_{j,c,\ell_1,m_1}^{[k-1]} \tag{4.45}$$

where C represents the Clebsch-Gordan coefficients and the output rotational order ℓ_3 and index m_3 are constrained in the typical way, i.e. $|\ell_1 - \ell_2| \leq \ell_3 \leq \ell_1 + \ell_2$ and $-(m_1 + m_2) \leq m_3 \leq m_1 + m_2$. This convolution is the main component of the interaction block, which generates mixed tensor features of the neighboring atoms to the central atom i, which defines the so-called "environment" of atom i. We note that here we suppress the tensor feature parity index, where it is assumed that

$p_3 = p_1 p_2$. Not detailed here is how these generated mixed tensor features defining the environment of atom i at layer k are the combined with the previous atomic features of the central atom itself $V_{i,c,\ell,m}^{[k-1]}$ to generate the new representation of atom i at layer k, $V_{i,c,\ell,m}^{[k]}$.

We forgo these details and focus on the larger picture. After the initial embedding and N_k interaction blocks, there will be defined a final representation for each atom i, $V_{i,c,\ell,m}^{[N_k]}$. This is a learned representation of the atom and its interactions with its environment. The scalar ($\ell = 0$) components are fed into a readout layer which learns an atomic energy E_i as seen below:

$$E_i = \text{MLP}_{\text{out}}(V_{i,c,\ell=0,m=0}^{[N_k]}) \tag{4.46}$$

Finally, the pair energies are summed over to define the total system energy, as follows:

$$E_{\text{system}} = \sum_i E_i \tag{4.47}$$

It is important to note that by definition as a message passing graph network, each atom's representation after each interaction block contains an increasingly larger "effective radius" for its neighborhood. We see that after one message passing layer, every atom's representation will now contain information regarding its immediate neighbors. After another, it will therefore contain information about not only its immediate neighbors, but their neighbors as well. After N_k message passing layers, it will "know" about neighbors that are N_k "hops" away. As a result, we find that as the number of message passing layers increases, representations become more complex due to repeated tensor product operations with neighboring representations and the computational workload increases. This limitation of message passing networks was one of the motivating factors for improving upon this architecture through

development of Allegro.

4.3.2 Allegro Architecture

Allegro [79] was designed to improve upon the successful NequIP architecture by removing the atom centered message passing construction and replacing it with a pair centered representation without message passing, with explicitly local interactions with neighboring pairs within a specified cutoff radius, r_{cut}. Additionally, for efficiency purposes, the Allegro architecture utilizes a "two-track" scalar and tensor feature definition. A diagram of the architecture is reproduced below in Figure 4.4:

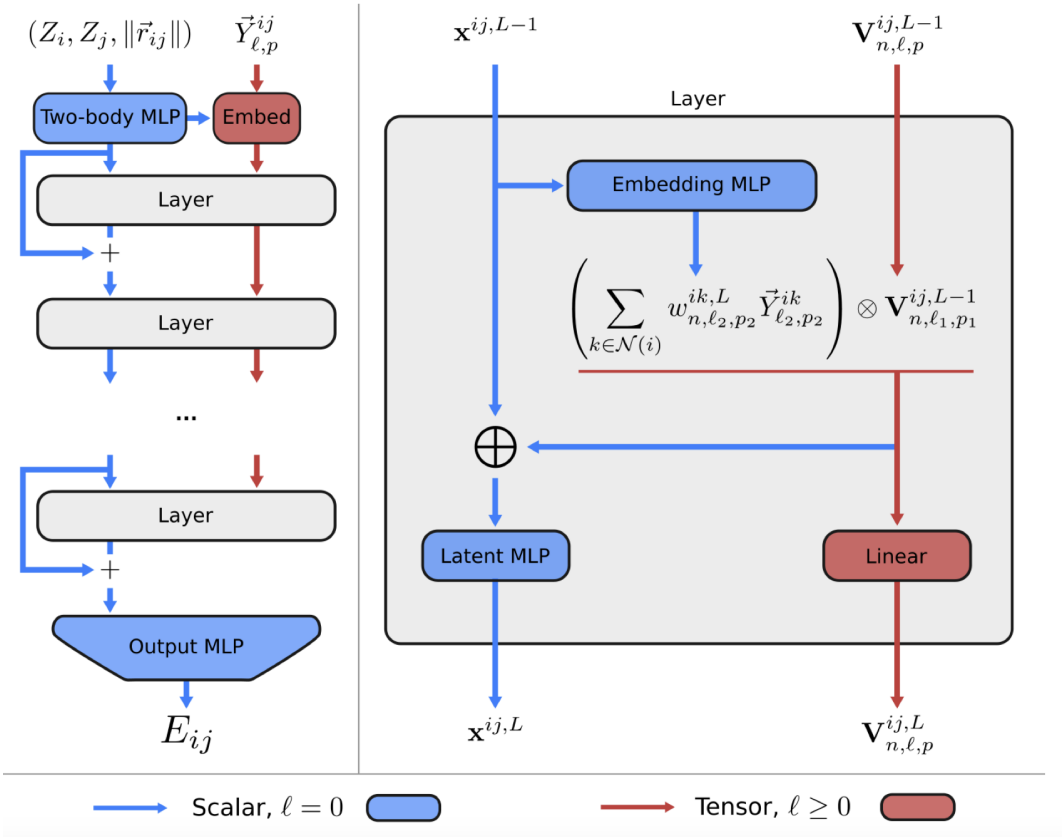


Figure 4.4: Visualization of the Allegro architecture as reproduced from reference [79] with permission.

The initial scalar feature embedding $\mathbf{x}^{ij,L=0}$ and tensor feature embedding components

$\mathbf{V}^{ij,L=0}$ are given below:

$$\mathbf{x}^{ij,L=0} = \text{MLP}_{\text{two-body}}\left(Z_i||Z_j||\mathbf{B}(r_{ij})\right) \cdot u(r_{ij}) \tag{4.48}$$

$$\mathbf{V}^{ij,L=0} = w_{n,\ell}^{ij,L=0}\overrightarrow{\mathbf{Y}}_{\ell}^{ij} \tag{4.49}$$

where the Z_i, Z_j are the atomic numbers of the atomic i and j which comprise the $i-j$ atomic pair, $\mathbf{B}(r_{ij})$ is the set of radial basis functions as defined in the NequIP formalism, $u(r_{ij})$ is the polynomial envelope function as defined as f_{env} in the NequIP formalism, and $\overrightarrow{\mathbf{Y}}_{\ell}^{ij}$ is the projection of r_{ij} onto the ℓ-th real spherical harmonic, where it is implied the components of this vector represent the order of the spherical harmonic m. Additionally, there is the learned coefficient $w_{n,\ell}^{ij,L=0}$, defined below:

$$w_{n,\ell}^{ij,L=0} = \text{MLP}_{\text{embed}}^{L=0}(\mathbf{x}^{ij,L=0})_{n,\ell} \tag{4.50}$$

Once the initial embedding has occurred, the scalar and tensor features are updated through several Allegro update layers, which function similarly to NequIP's interaction blocks. Here, spherical harmonic projections are constructed which represent neighboring ik pairs in the local environment of the central ik pair. These environment representations are combined with the tensor features of the ij pair through the tensor product, which generates mixed features relating the ij and ik pairs. The outputs from this tensor product are quite similar to the convolutions which are defined in the NequIP architecture. The products are defined below:

$$\mathbf{V}_{n,(\ell_1,\ell_2)\to(\ell_3)}^{ij,L} = \sum_{k\in\mathcal{N}(i)} w_{n,\ell_2}^{ik,L}\left(\mathbf{V}_{n,\ell_1}^{ij,L-1} \otimes \overrightarrow{\mathbf{Y}}_{\ell_2}^{ik}\right) \tag{4.51}$$

where $w_{n,\ell_2}^{ik,L}$ is again a learned coefficient. Here again, we are suppressing the parity index and it is again assumed that $p_3 = p_1 p_2$. It is important to note that although

this product is very similar to the NequIP convolution, due to the pair construction in Allegro, this is not a message passing operation, and the "effective radius" does not increase. The tensor mixing between the ij pair and its ik pair neighbors remains strictly local and within the radius r_{cut}. After the tensor product is computed, the scalar and tensor output components are split into the "two-track" design to update the scalar and tensor feature vectors. The scalar feature vector update is defined below:

$$\mathbf{x}^{ij,L} = \text{MLP}^{L}_{\text{latent}} \left(\mathbf{x}^{ij,L-1} || \bigoplus_{\ell_1,\ell_2} \mathbf{V}^{ij,L}_{n,(\ell_1,\ell_2)\to(\ell_3=0)} \right) \cdot u(r_{ij}) \tag{4.52}$$

Here what this means is that all tensor product outputs which are scalars, i.e. $\ell_3 = 0$, are concatenated along with the previous layer's scalar feature vector $\mathbf{x}^{ij,L-1}$ to serve as the input to a neural network which then learns the updated scalar feature vector. This learned function is then multiplied by the envelope function to ensure it remains smooth and that it goes to zero at r_{cut}. The tensor feature vector is updated by linearly combining all tensor product outputs of the same degree scaled by learned weights:

$$\mathbf{V}^{ij,L}_{n,\ell} = \sum_{n',(\ell_1,\ell_2)} w_{n,n',(\ell_1,\ell_2)\to(\ell)} \; \mathbf{V}^{ij,L}_{n',(\ell_1,\ell_2)\to(\ell)} \tag{4.53}$$

After all N_L of the Allegro update layers have been computed, the final scalar features are fed into the output layer, which defines the pair energy as follows:

$$E_{ij} = \text{MLP}_{\text{out}}(\mathbf{x}^{ij,N_L}) \tag{4.54}$$

Finally, the readout layer defines the total energy of the system as the sum of the learned pair energies as follows:

$$E_{\text{system}} = \sum_i \sigma_{Z_i} E_i + \mu_{Z_i} = \sum_i \sigma_{Z_i} \left(\sum_{j \in \mathcal{N}(i)} \sigma_{Z_i, Z_j} E_{ij} \right) + \mu_{Z_i} \tag{4.55}$$

where σ_{Z_i} and σ_{Z_i,Z_j} are species specific scaling factors and μ_{Z_i} is a species specific shift parameter. Note that there are some intricacies not elucidated here, such as how they more efficiently compute the tensor product and explicitly how they intelligently handle the tensor product output "paths". Many of these details are not necessary to understand the general concepts within the framework for the purpose of utilizing the method in our research, and are more suited to understanding the programming details and clever efficiency improvements in the code. These details coupled with the strictly local construction serve to greatly improve the efficiency compared to NequIP without loss of accuracy.

One excellent result from their demonstration that the Allegro model potential was able to simulate lithium phosphate Li_3PO_4 and the resultant MSD curve was in fairly good agreement with the MSD curve based on FPMD calculations, which is reproduced in Figure 4.5.

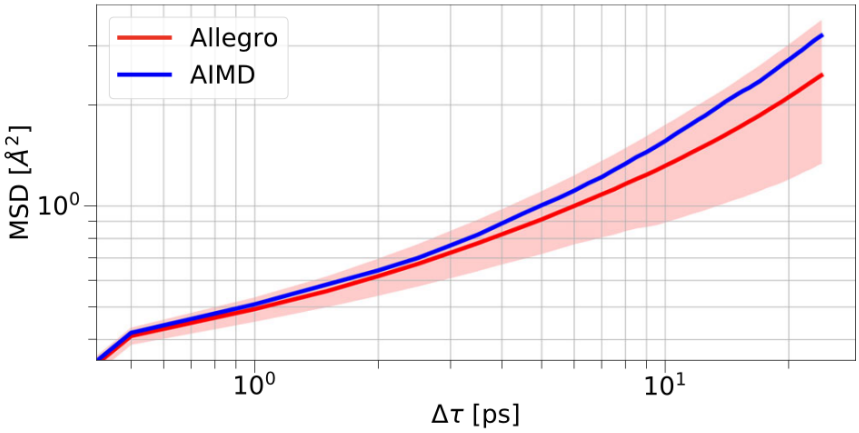


Figure 4.5: Comparison of Li_3PO_4 MSD results between an Allegro potential model and FPMD, as reproduced from reference [79] with permission.

The excellent results and efficiency of the Allegro software package inspired us to investigate this model for our our purposes in collaboration with one of the current programmers, Chuin Wei Tan, as presented in section 5.2.

Chapter 5: Research Project Summaries

5.1 Computational investigation of the structural and electrolyte properties of the extended family of lithium (thio)boracite materials: $Li_4B_7O_{12}Cl$ and beyond

This section outlines my contributions to the publication contained in Appendix B, which primarily includes structural investigations and phonon calculations of the 8 boracite and thioboracite materials discussed in section 1.2.

5.1.1 Optimized Static Lattice $R3c$ Structures

Finding the optimized static lattice ground-state structures for the family of extended (thio)boracites was one area that I focused on during the collaboration. Dr. Li and Dr. Holzwarth's previous work [14] found that the $Li_4B_7O_{12}Cl$ material has a rhombohedral $R3c(161)$ ground-state structure that is a distortion of the well documented $F\bar{4}3c(219)$ cubic structure. The lattice framework is composed of BO_3 planar triangular and BO_4 tetrahedral pieces with distinct void sections. The void sections contain a central Cl halogen which is surrounded by lithium sites. In the cubic conventional cell setting, these lithium sites are denoted by $32e$ Wyckoff positions with 25% fractional occupancy and fully occupied $24c$ Wyckoff positions. These lithium sites map to fully occupied $4 \times 8a$ Wyckoff positions and unoccupied $3 \times 8b$ Wyckoff positions in the rhombohederal conventional cell. The empty b sites form so-called natural interstitial sites which are metastable lithium positions that are utilized during ion migration in a concerted ion "hop" involving 2 lithium ions. This mechanism is presented in more depth in that paper and will not be elaborated here.

Building upon this previous investigation, we took the $R3c$ $Li_4B_7O_{12}Cl$ structure

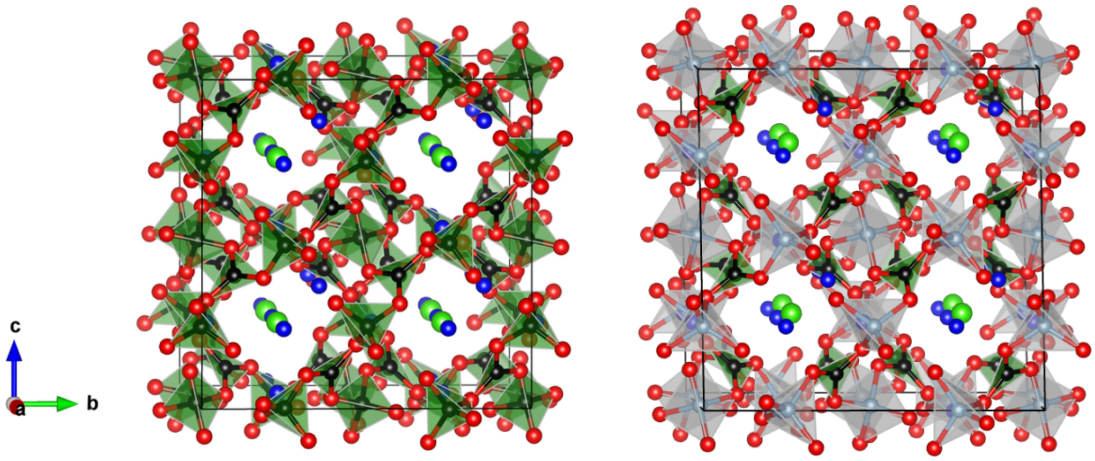


Figure 5.1: $R3c$ distorted conventional cell structures for the $Li_4B_7O_{12}Cl$ (left) and $Li_4Al_3B_4O_{12}Cl$ (right).

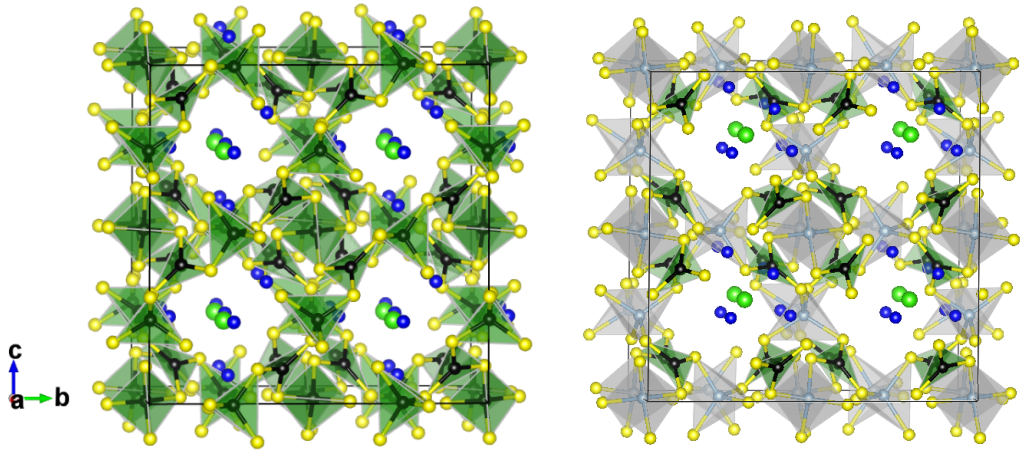


Figure 5.2: $R3c$ distorted conventional cell structures for the $Li_4B_7S_{12}Cl$ (left) and $Li_4Al_3B_4S_{12}Cl$ (right).

and substituted the oxygen sites with sulfur. Performing structural relaxation calculations allowed us to find an optimized ground-state candidate structure for $Li_4B_7S_{12}Cl$. For the aluminum Li_4 materials, this wouldn't be such a simple exercise. We needed to attempt to find a similar $R3c$ distortion for $Li_4Al_3B_4O_{12}Cl$ and $Li_4Al_3B_4S_{12}Cl$. In order to do this, we needed to take the $R3c$ structures for $Li_4B_7O_{12}Cl$ and $Li_4B_7S_{12}Cl$ and determine how to partially substitute boron with aluminum in a $4:3$ ratio, while maintaining the $R3c$ symmetry. In the $R3c$ conventional cell, boron atoms are found

in fully occupied $8a$ and $2 \times 24b$ Wyckoff positions. This means there are fortunately only 2 possible ways to partially substitute boron for aluminum in this specific ratio while maintaining the symmetry - by choosing 1 of the 2 groups of $24b$ boron Whyckoff positions to substitute with aluminum. After relaxing the possible configurations in QE, we were able to find the lowest energy partial substitution with the correct $R3c$ symmetry. In Figure 5.1 we see the distorted $R3c$ structures for the $Li_4B_7O_{12}Cl$ and $Li_4Al_3B_4O_{12}Cl$ materials, and in Figure 5.2, we see the distorted $R3c$ structures for the $Li_4B_7S_{12}Cl$ and $Li_4Al_3B_4S_{12}Cl$ materials. Static lattice energies can be found in Table I in Appendix B.

For the $Li_6B_7S_{13}Cl$ material, we had to start with the $Li_6B_7S_{13}I$ $F\bar{4}3c$ structure from the experimental investigation and substitute the iodine with chlorine. In the cubic structure, we have a different lithium substructure compared to the original boracite - here we have fully occupied $32e$ Wyckoff positions and $66.\bar{6}\%$ fractionally occupied $24c$ Wyckoff positions in the conventional cell. Attempting to find a similar $R3c$ structure with ordered lithium sites resulted in a low energy structure more similar to the boracite - an $R3c$ conventional cell structure with fully occupied $24c$ Wyckoff positions and 75% occupied $32e$ Wyckoff positions. Once we determined a suitable $R3c$ structure for $Li_6B_7S_{13}Cl$, we were able to employ similar searches to those done previously to find the remaining substituted Li_6 structures - $Li_6B_7O_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$. These $R3c$ structures for the Li_6 materials are shown in Figures 5.3 and 5.4.

5.1.2 Optimized Static Lattice Cc Structures

During the investigation of the $Li_6B_7S_{13}Cl$ $R3c$ structure, a new structure with lower static lattice energy than the $R3c$ structure was found with monoclinic $Cc(9)$ symmetry. Once this structure was found, we then attempted to use this as the ba-

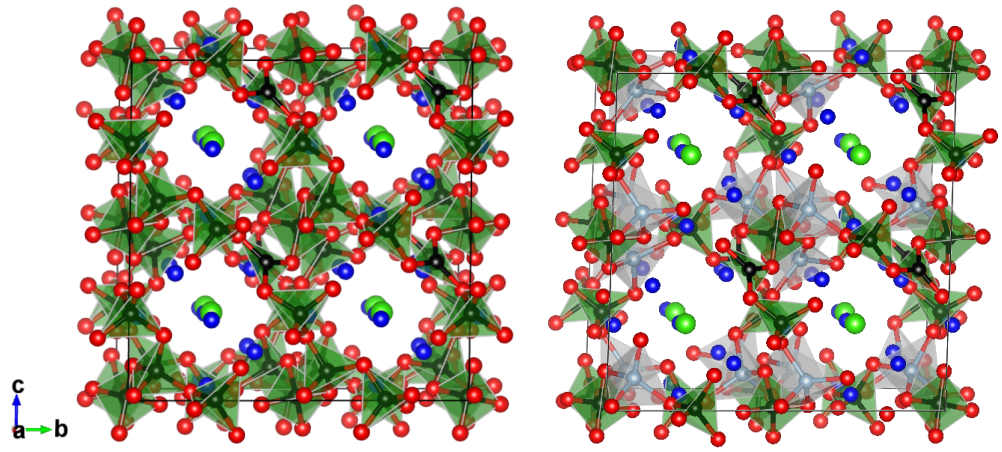


Figure 5.3: $R3c$ distorted conventional cell structures for the $Li_6B_7O_{13}Cl$ (left) and $Li_6Al_3B_4O_{13}Cl$ (right).

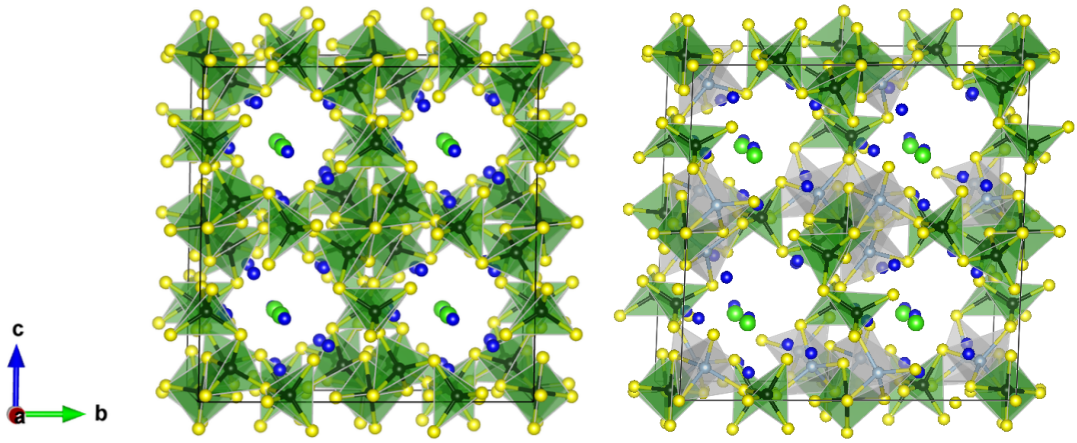


Figure 5.4: $R3c$ distorted conventional cell structures for the $Li_6B_7S_{13}Cl$ (left) and $Li_6Al_3B_4S_{13}Cl$ (right).

sis for a search for other possible Cc structures for the remaining Li_6 materials. This search yielded a new Cc structure for each of the remaining Li_6 materials - $Li_6B_7O_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$. It was also found that all of the Li_6 Cc structures have lower static lattice energies than their $R3c$ counterparts except one - $Li_6B_7O_{13}Cl$. In Figure 5.5 we see the comparison between the lower energy Cc $Li_6B_7S_{13}Cl$ structure compared to its $R3c$ counterpart, and in Figure 5.6 we see the lower energy Cc configurations of $Li_6Al_3B_4O_{13}Cl$ and $Li_6Al_3B_4S_{13}Cl$.

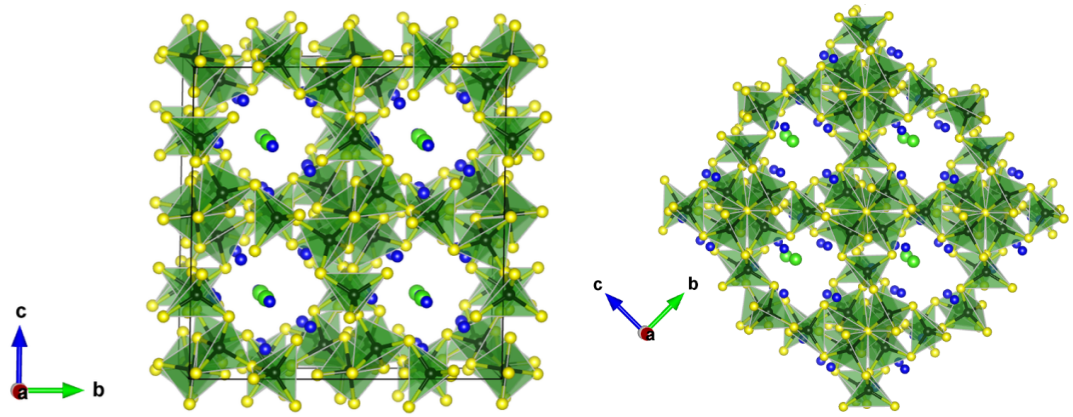


Figure 5.5: Conventional cell structures for $Li_6B_7S_{13}Cl$ in $R3c$ (left) and Cc (right) configurations.

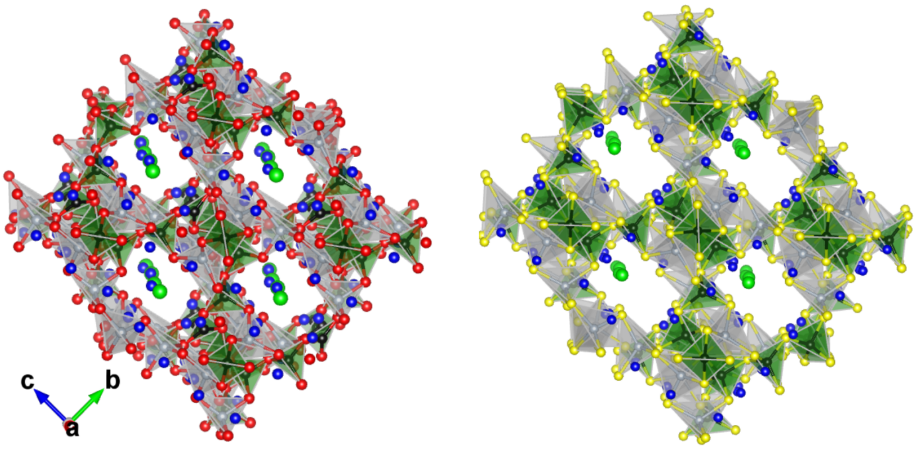


Figure 5.6: Cc conventional cell structures for the $Li_6Al_3B_4O_{13}Cl$ and $Li_6Al_3B_4S_{13}Cl$ configurations.

We then desired to investigate whether or not a similar monoclinic structure could be found for the Li_4 materials. To do so, we investigated how we could modify the existing Li_6 structure to facilitate the Li_4 stochiometry while maintaining the Cc symmetry. In the primitive cell, this involves determining how one can remove 2 units of Li_2O/Li_2S from the cell - in other words, 4 lithium atoms and 2 oxygen/sulfur atoms. In the Cc primitive cell, there is only 1 Wykcoff position present, which happens to contain 2 symmetry related positions. This means we need to simply find 1 pair of

symmetry related oxygen/sulfur and 2 pairs of symmetry related lithium to remove. For the oxygen/sulfur, we carefully compared the experimental $F\bar{4}3c$ structure and the newly found Cc structure, and determined that extra pair of oxygen/sulfur atoms in the cubic Li_6 primitive cell are positioned in $2a$ sites that are not present in the Li_4 cubic primitive cell. We then found the analogous extra pair of oxygen/sulfur in the Li_6 Cc structure and removed them. For the 2 pairs of lithium atoms, we simply did a search of the possible ways of choosing 2 pairs out of the 6 pairs present in the Cc primitive cell to remove - resulting in 15 possible initial configurations for the Li_4 Cc material. This search was *not* exhaustive, but was surely systematic. The Cc search resulted in similar Cc structures for all of the Li_4 materials - but they all had *much* higher static lattice energies than their corresponding $R3c$ and $F\bar{4}3c$ structures, and were not considered for further investigation.

5.1.3 Optimized Static Lattice $F\bar{4}3c$ Structures

For the original $Li_4B_7O_{12}Cl$ boracite, the disordered cubic structure cannot be ordered while maintaining the symmetry - it distorts upon choosing specific lithium positions for the fractionally occupied lithium sites. In order to maintain the $F\bar{4}3c$ symmetry, one would need to find possible Wyckoff positions for the lithium ions such that they are all fully occupied. In this case, there are 32 lithium ions in the conventional cell - 24 of which are in fully occupied $24c$ Wyckoff positions. This leaves 8 lithium positions with which to find Wyckoff positions for. The possible Wyckoff labels that contain 8 symmetry related sites in the cubic setting are the $8a$ and $8b$ sites - which are special sites at $(0, 0, 0)$ and $(1/4, 1/4, 1/4)$. The b sites are already occupied with Cl atoms and the a sites were determined to be unlikely lithium positions due to the fact that they are embedded in the framework and the lithium migration occurs in and through the void regions. The only remaining sensible choice then

would be to place all of the lithium in 32e sites and leave the 24c sites unoccupied. This static lattice configuration was found to be energetically unfavorable compared to the distorted $R3c$ structure in previous research done by Dr. Li and Dr. Holzwarth. Here we wanted to see if we could find similar ordered cubic structures for the new (thio)boracite substitutions and compare them energetically to their $R3c$ counterparts. We were able to perform similar analysis for the Li_4 materials and indeed found ordered cubic configurations for these materials. It was found that there was one material where the cubic structure was in fact energetically favorable compared to the $R3c$ structure - $Li_4B_7S_{12}Cl$. The remaining cubic Li_4 materials had much higher static lattice energies than their corresponding $R3c$ structures. In Figure 5.7 we see the comparison between the lower energy ordered $F\bar{4}3c$ $Li_4B_7S_{12}Cl$ structure compared to its $R3c$ counterpart.

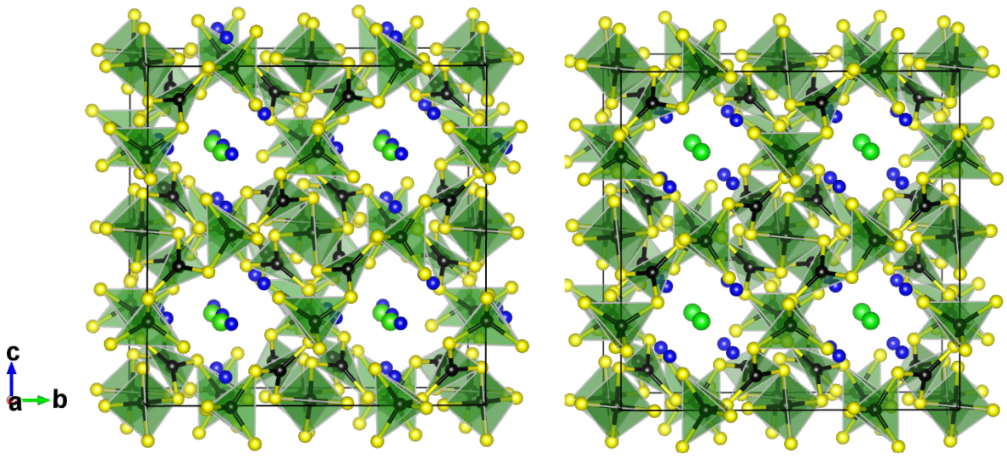


Figure 5.7: Conventional cell structures for $Li_4B_7S_{12}Cl$ in $R3c$ (left) and ordered $F\bar{4}3c$ (right) configurations.

For the Li_6 materials however, there are more lithium ions present, and therefore the possible Wyckoff labels available for lithium occupation are quite different. In this case, there are now 48 lithium positions in the conventional cell, and there are 3 choices of where to place the lithium ions to maintain an ordered $F\bar{4}3c$ structure. The

$24c$ lithium sites can be left alone and the remaining 24 lithium could fill $24d$ sites, or the $24c$ sites can be emptied so to speak, and the lithium be placed in either $48f$ or $48g$ sites. For the first option, this is not possible because the $24d$ sites again are special sites that are already occupied with B atoms. This leaves the possibility of finding an ordered $F\bar{4}3c$ structure for the Li_6 materials by sampling the space of possible f and g sites that can be fully occupied with lithium. This search was done on somewhat course grids of possible positions, namely $x = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. Ordered cubic structures were in fact found for all of the Li_6 materials as well. In this case none of them were energetically favorable compared to their $R3c$ counterparts - they all have much higher static lattice energies, and were not considered for further investigation.

5.1.4 Phonon Calculations

While the phonons of several materials have previously been calculated with ABINIT, in this work we have calculated the phonons for all of the extended (thio)boracites using Phonopy. For each material, the band structures and phonon density of states (PDOS) were calculated with the same settings within QE and Phonopy. Displacement supercells generated by Phonopy were generated from the primitive cell using $2 \times 2 \times 2$ scaling, where the displacements themselves were set using the Phonopy default value of $0.01\mathring{A}$. The SCF calculations in QE were calculated using planewave energy cutoffs of $|\mathbf{k} + \mathbf{G}|^2 \leq 81Ry$ and a k-point grid of $1 \times 1 \times 1$. The coarse k-point sampling should be sufficient due to the fact that the lattice volumes are so large. The band structures were interpolated from a dataset of calculated datapoints with a density of 101 sampling points per path. For the PDOS, a gamma centered mesh with a $16 \times 16 \times 16$ density grid was used. Helmholtz free energy was calculated along the temperature range from $T = 0$ to $T = 1000K$ in intervals of $10K$. The resulting combined band structure and PDOS plots are given in Figures 5.8, 5.9, 5.10,

and 5.11 for the lowest energy structures of each material in the (thio)boracite family investigated.

We see here that the lowest energy structure for each material has real phonon frequencies, which suggests they have dynamic stability, as least within the confines of the frozen phonon approximation given here. Not shown in this section, but detailed in Appendix B, are the band structure and PDOS plots for some of the higher energy material structures. Interestingly, the $Li_6B_7S_{13}Cl$ $R3c$ structure was found to have some imaginary frequencies, which is in contrast to the experimental results by Kaup et al. [20]. This may or may not be a result of the different choice of halogen - they after all investigated iodine whereas we are using chlorine.

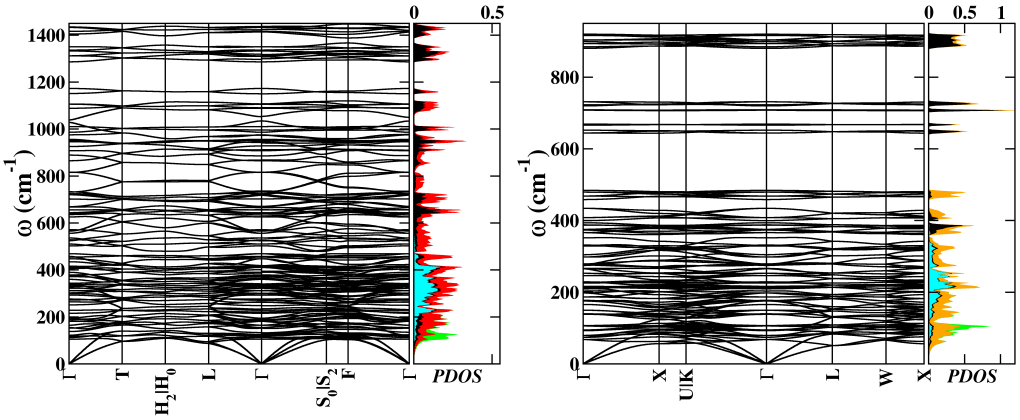


Figure 5.8: Band structure and PDOS plots for the $Li_4B_7O_{12}Cl$ $R3c$ configuration (left) and the $Li_4B_7S_{12}Cl$ $F\bar{4}3c$ configuration (right).

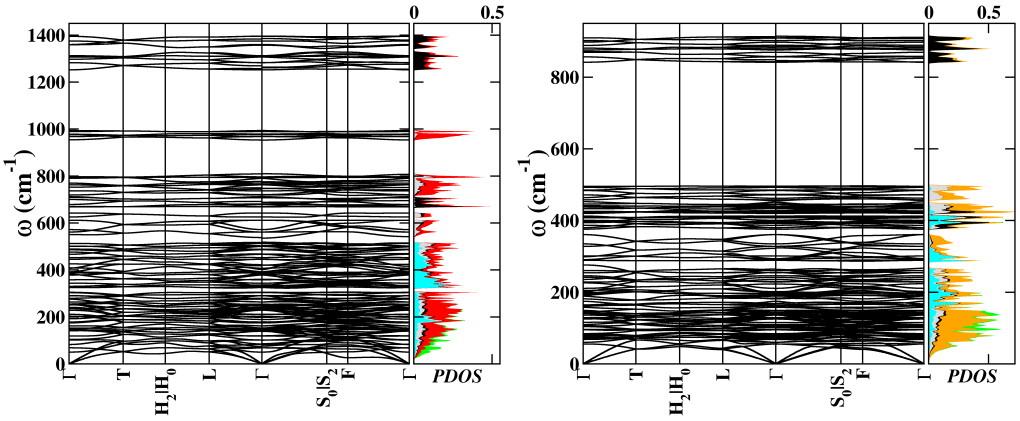


Figure 5.9: Band structure and PDOS plots for the $Li_4Al_3B_4O_{12}Cl$ $R3c$ configuration (left) and the $Li_4Al_3B_4S_{12}Cl$ $R3c$ configuration (right).

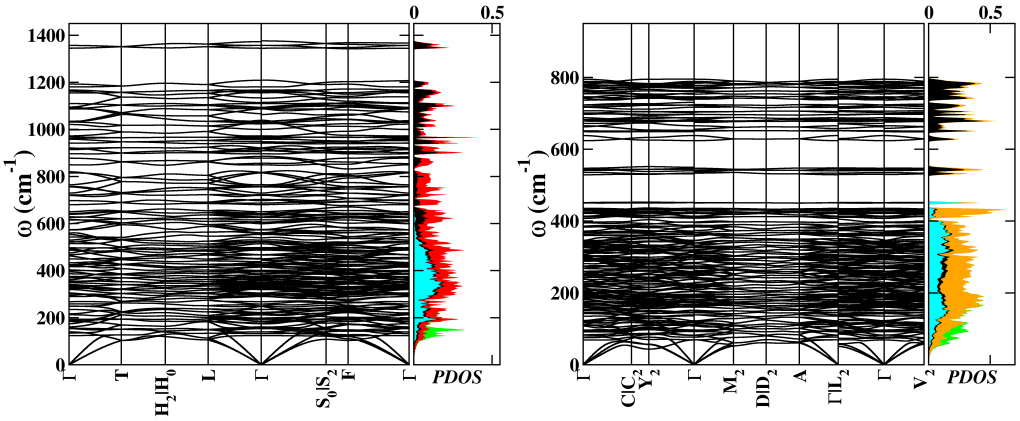


Figure 5.10: Band structure and PDOS plots for the $Li_6B_7O_{13}Cl$ $R3c$ configuration (left) and the $Li_6B_7S_{13}Cl$ Cc configuration (right).

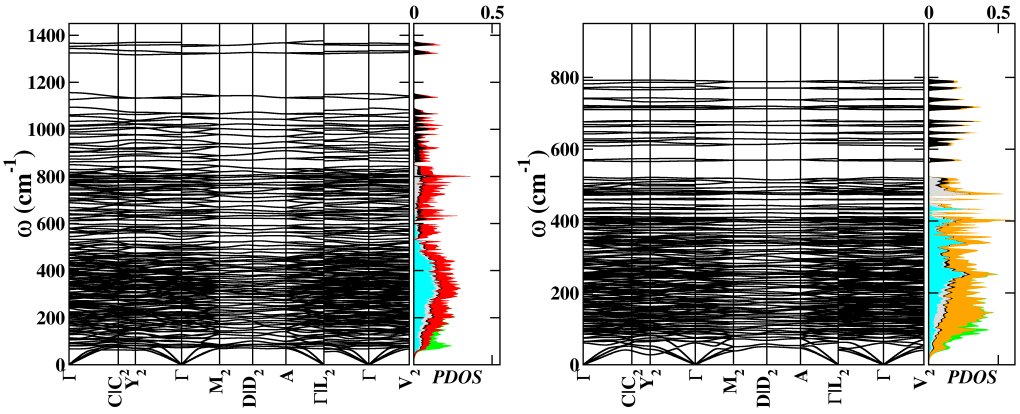


Figure 5.11: Band structure and PDOS plots for the $Li_6Al_3B_4O_{13}Cl$ Cc configuration (left) and the $Li_6Al_3B_4S_{13}Cl$ Cc configuration (right).

5.2 A study of the use of machine learning methodology for computing ionic conductivity in an example solid-state electrolyte material

This section details some of the successes and challenges faced when assessing the capability of the Allegro machine learned interatomic potential architecture [79]. The choice to utilize this machine learned potential was made to extend first principles capabilities: using first principles molecular dynamics alone, it is not feasible to run simulations for long enough time scales to get good statistics for calculating the full conductivity, and there are also limitations on system size and target temperature as well. To test the Allegro software, we used an alloy of lithium phosphate (Li_3PO_4) and lithium orthosilicate (Li_4SiO_4) for our system, based on works by Deng et al. [23], which is discussed in section 1.3. This system was chosen based on the fact that it is a simple system with small unit cell and low number of atoms, as well as being a good solid electrolyte material that has been well studied. This section serves as a summary of the preprint contained in Appendix C.

5.2.1 Test System and Initial Allegro Familiarization

During an initial familiarization with the Allegro software and machine learning methodology, we began to systematically investigate the training settings and hyperparameter choices within the software for optimal training, in the interest of balancing accuracy of the potential model and computational time and resource considerations. For this testing, we constructed two structures of $Li_{3.25}Si_{0.25}P_{0.75}O_4$, which we call "structure S1" and "structure S2". For the initial hyperparameter investigation, only structure S1 was used. The structures are loosely based off of work by Deng et al. [23], with a $Li_3PO_4 : Li_4SiO_4$ ratio of 3 : 1, and are shown in Figure 5.12 below.

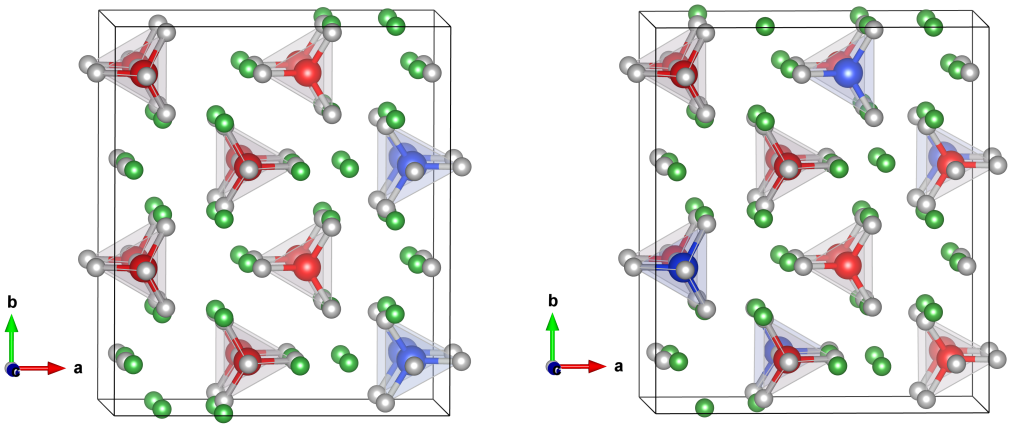


Figure 5.12: Figure depicting the relaxed electronic ground-state structures for two configurations of $Li_{3.25}Si_{0.25}P_{0.75}O_4$: structure S1 (left) and structure S2 (right). Green, blue, red, and white spheres represent Li, Si, P, and O atoms respectively.

For the hyperparameter testing, we focus on five main parameters that impact training: the cutoff radius r_{cut}, the maximum rotational order of the tensor representations ℓ_{max}, the number of tensor representations n_{tensor}, the number of scalar representations n_{scalar}, and the dimensions of the various embedded learned functions $\dim_{\text{MLP}} = w \times d$. Preliminary testing showed that an r_{cut} of 6.0 $\mathring{A}$ was a good choice for this system - increasing r_{cut} beyond 6.0 $\mathring{A}$ resulted in negligible accuracy improve-

ments at large computational cost. It was clear from this investigation that having r_{cut} too small negatively impacts the accuracy of the model substantially, as seen in Figure 5.13.

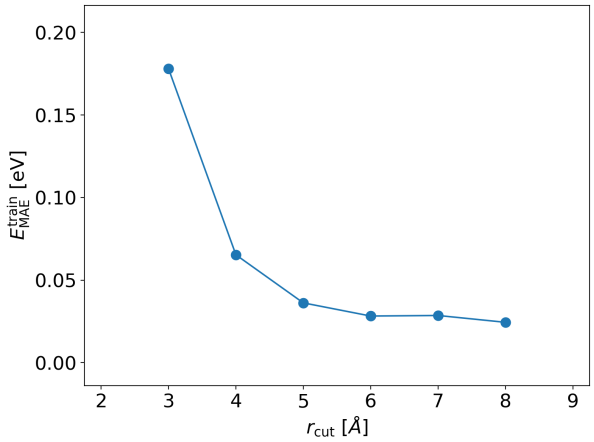


Figure 5.13: Figure depicting the impact of the choice of r_{cut} on the model's training E_{MAE} error for a fixed model "size" (choice of hyperparameters) and input dataset size (number of configurations). Here r_{cut} is given in $\mathring{A}$ and E_{MAE} is given in eV. E_{MAE} is defined in equation 53 of the preprint manuscript C.

We note that this r_{cut} testing was rather coarse, as dataset split and model initialization seeds were kept fixed, and dataset size and the impacts of other hyperparamters choices were not considered. Nevertheless, this choice of r_{cut} is in line with research done using similar materials [79] and seems reasonable given the geometry of the system. After determining the choice of r_{cut}, we then tested the three remaining hyperparameters of interest by defining "small", "medium", and "large" model sizes with hyperparameters defined in Table 5.1 below.

In order to test the model hyperparameter choices, we must first determine the dataset size to use for training. For a given model size, it has some fixed capacity to learn associations between inputs and outputs. If the dataset is too small, there will be fewer associations to be learned, and one might expect a relatively "small" model would be sufficient to learn all of these associations and there would be no need to

Table 5.1: Table defining Allegro model "sizes", which determine the specific hyperparameter choices for the model.

	small	medium	large
r_{cut}	6.0	6.0	6.0
l_{max}	1	2	3
n_{tensor}	16	32	64
n_{scalar}	32	64	128
MLP_{dims}	1×32	2×64	3×128
n_{params}	25K	91K	387K

have a larger model. For this reason, to test the differences between model sizes, we need a large enough dataset with enough associations present to fill the capacity of the model. Once we have a large enough dataset to fill this capacity, we can see how changing the hyperparameters will impact the accuracy and computational resource and time requirements in order to find an ideal balance.

For this test, we looked at a "medium" model and tested dataset sizes of 2500, 5000, and 7500. These serve as input datasets that are randomly split internally into training and validation datasets at a ratio of 80 : 20. Three models with different dataset split seeds were trained for each dataset size test and averaged to somewhat account for random noise present in training. Due to time constraints, we kept the model initialization seeds fixed, though previous experience suggests these are of lower impact than the dataset split. The results are shown in Figure 5.14 below.

From this assessment it was determined that 5000 configurations - 4000 training and 1000 validation - was a good choice for our test system. We note that due to the fact that the input data is generated from molecular dynamics trajectories, there may be correlated frames contained in both training and validation subsets. As a result, we expect the validation results to primarily characterize in-distribution interpolation and should not generalize to independent trajectories, temperatures, structures, or materials.

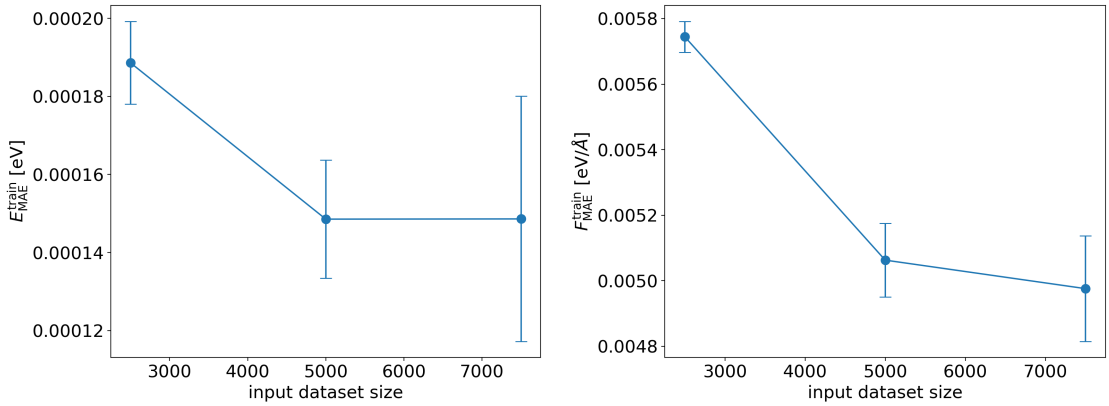


Figure 5.14: Dataset size test with 2500, 5000, and 7500 input dataset sizes, showing average training E_{MAE} (left) and F_{MAE} (right) error vs. input dataset size (number of configurations), with error defined by the standard deviation. Here E_{MAE} is given in eV and F_{MAE} is given in eV/$\mathring{A}$. F_{MAE} is defined in equation 54 of the preprint manuscript C.

Given the determination of dataset size, we then proceeded to test the model sizes defined above, again using three dataset split seeds and keeping the model initialization seeds fixed. The average results for each model size is given in Figure 5.15 below.

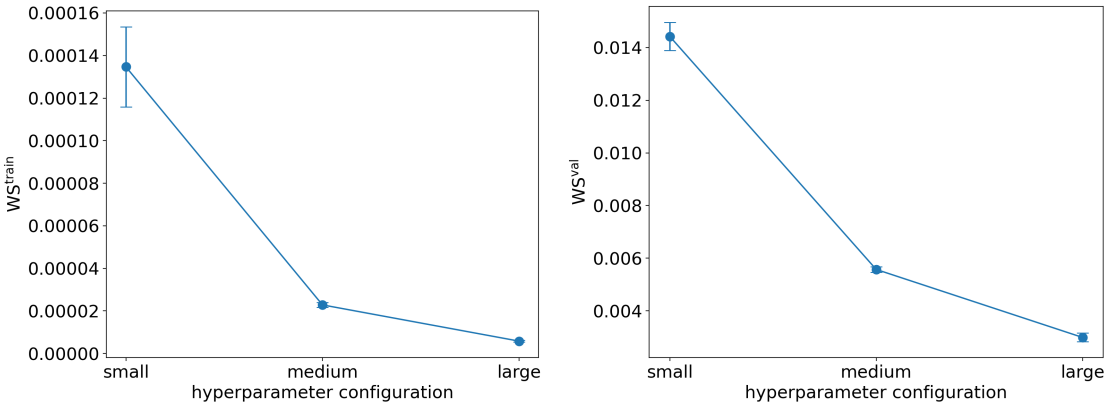


Figure 5.15: Hyperparameter test with "small", "medium", and "large" model sizes, showing average training (left) and validation (right) weighted sum values vs. model size, with error defined by the standard deviation. The weighted sum values are unitless, and are defined in equation 56 of the preprint manuscript C.

We see from these results that although the large model has the lowest error, the

improvement from the medium model size is much lower than the improvement from small to medium, and in fact the computational resources needed and time required makes the large model size impractical. Furthermore, while the improvement from small to medium seems impressive, at this scale, these training and validation errors are so small that these models behave somewhat similarly when using them to run simulations in LAMMPS. For our testing, we decided to primarily investigate small models, with a few medium models added for comparison.

Hyperparameter and dataset testing detailed here, as well as other training settings not mentioned such as choice of optimization algorithm, early stopping criteria, learning rate decay, and so on, were informed greatly through very helpful discussions with Dr. Fan Yang and graduate student Chuin Wei Tan of the Materials Intelligence Research group at Harvard University, who is now a collaborator on this project. We note that this is a limited empirical study of possible dataset sizes and hyperparameter choices based in part on default settings and suggestions provided by the developers, as well as previous preliminary testing not shown here, and could in principle be expanded upon with more testing to find globally optimal choices.

5.2.2 Molecular Dynamics Stability Issues

Once we determined our hyperparameter and training settings, we began looking at training several small and medium models with datasets containing a random sampling of configurations generated from first principles molecular dynamics. One decision that was made was to include both structures 1 and 2 in the training, which we call "combined" training. The molecular dynamics simulations were ran at temperatures of 100K, 400K, 600K, and 900K, where they were sampled at a ratio of $1:1:2:2$ for each structure. These molecular dynamics samplings for each structure contained 5000 configurations. Of the 10,000 combined samplings of structure 1

and 2, 5000 were randomly sampled, ensuring our "combined" dataset weighed each structure evenly. The combined training is thought to be beneficial by increasing the "diversity" of the dataset, which should improve the model's ability to generalize to untrained configurations within the interpolative range defined by the input data. We additionally looked at two "single structure" models, trained on only structure S1 or S2, to see how they compare to the combined training models. This would allow us to determine if training with both structures negatively impacts the model's performance on one of the structures individually, in which case it would be better to train a model on a single structure only.

During the course of testing preliminary trained models, we ran into the interesting situation where models which seemed to be well trained with low errors would have random stability issues when using them in LAMMPS to run molecular dynamics simulations. This happens when, during the course of a molecular dynamics simulation, the atomic positions drift slightly outside of the trained "range" of the Allegro potential, and the model fails to accurately predict the forces and energies. This seems to occur when the atomic pairs get too close and are not correctly repelled due to the coulombic interaction. This issue turns out to be well known, with some work on mitigating this issue including the addition of a close-range repulsive potential [86]. This instability can be visualized as large temperature or energy spikes during the simulation that are unphysical, as shown in Figure 5.16 below, where we see example temperature plots.

For our study, we decided to instead attempt to systematically construct configurations with closer pair distances based on the equilibrium structure. Roughly 400 additional "close-pair" configurations were added to the 5000 configuration input datasets for each structure, resulting in a new combined input dataset of roughly 5800 configurations, or about 4640 training and 1160 validation configurations. We

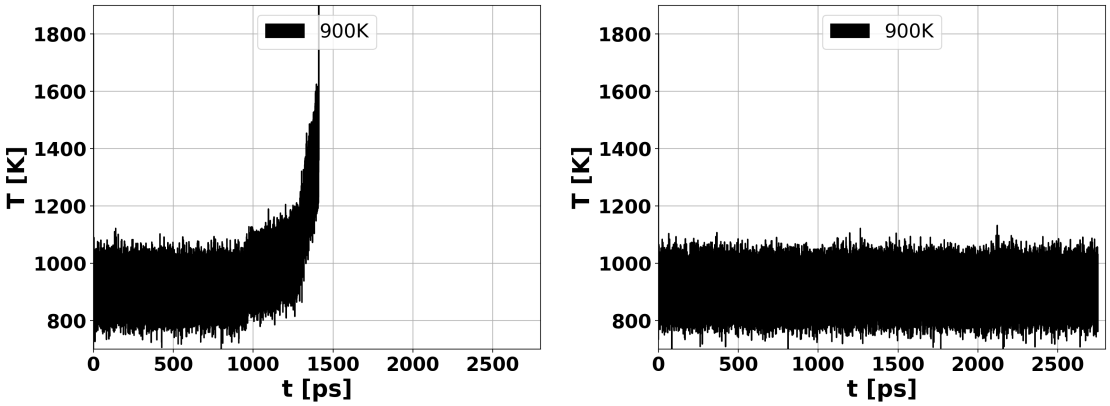


Figure 5.16: Figure showing example temperature plots, depicting model instability during a molecular dynamics run (left) compared to a stable molecular dynamics run (right), both at target temperatures of 900K.

also adopted a 900K "screening test" to determine whether or not a model exhibits this stability issue during a molecular dynamics simulation. If so, the model was not considered for molecular dynamics testing below. In our limited experience, this addition of close-pair configurations reduced the number of models that exhibit this instability during the 900K screening test. Regardless of the outcome of the screening, all models can be considered for comparison of training and validation errors as well a limited structural prediction test, which are detailed in the proceeding sections.

5.2.3 Allegro Model Analysis

At the end of this initial Allegro testing and after discovering and attempting to mitigate the random model instability by including constructed close-pair training configurations, we ended up training and analyzing ten Allegro models, as defined in Table 5.2 below.

The training behavior is given in Table III of the preprint in Appendix C, where we see that the training behavior is substantially different for the different models. Many of the models trained the full two weeks before running out of time, but others

Table 5.2: List of 10 different Allegro models studied in this work. Note that the labels serve as a compact notation for the model definition, in the format XYZ, where X is the input dataset label (C for "combined", S for "single"), Y is the model size label (S for "small", M for "medium" as defined in table 5.1), and Z is simple a numerical label for further model differentiation, such as different model initialization.

Label	Model Size	Training Structure(s)
CS1	Small	S1 + S2
CS2	Small	S1 + S2
CS3	Small	S1 + S2
CS4	Small	S1 + S2
CS5	Small	S1 + S2
CS6	Small	S1 + S2
CM1	Medium	S1 + S2
CM2	Medium	S1 + S2
SS1	Small	S1
SS2	Small	S2

converged quite quickly to a local minimum of the loss surface. We see the training and validation error metrics in Tables IV and V in the preprint in Appendix C.

It turns out that only two of the models - CS1 and CM2 - failed the 900K screening test. Interestingly, of the two models which reached a local minimum after less than 200 epochs and stopped training - CS1 and CS6 - we see that its not clear this "hurt" the model overall: exactly one passed and one failed the 900K screening test, and both have similar magnitudes of error. It's clear that the comparison of training and validation error is not the whole story regarding a model's "quality".

5.2.4 Assessment of Structural Prediction Capability

When running molecular dynamics simulations from equilibrium, the starting structure is a particular equilibrium structure. Since this structure isn't explicitly in the training set but is related to the training configurations, we decided to test the models' ability to interpolate to this untrained structure by seeing if the model can correctly predict it. To do this, we input the known equilibrium structure to LAMMPS, and

run a structural relaxation calculation using an Allegro potential. If the model is able to accurately predict this as the equilibrium structure, it should not relax at all. The more the structure changes, which can be quantified by a change in volume or total energy, the less accurate the model is in predicting that equilibrium structure. The results are given in Table VI in the preprint in Appendix C.

Given these results, we can see if there is a relationship between the validation error and the ability of the model to predict the equilibrium structure, as defined by a volume or energy change after relaxation in LAMMPS. This is shown in Figure 5.17 below.

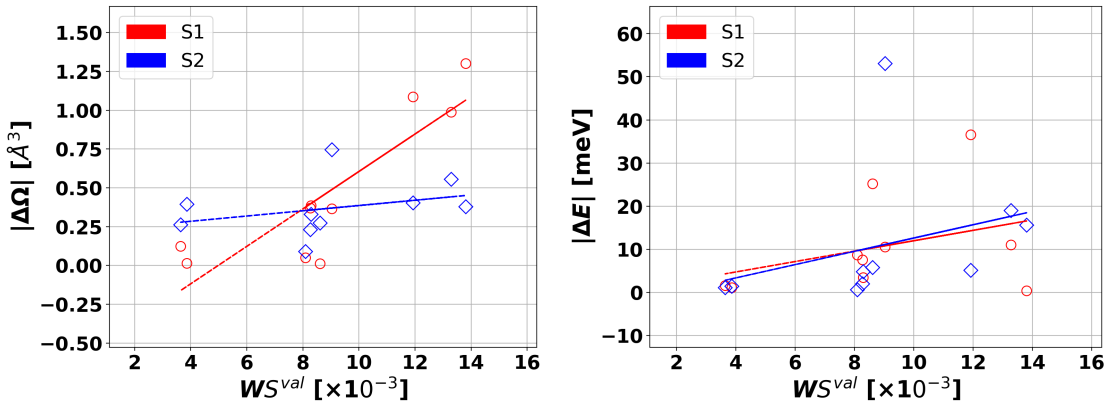


Figure 5.17: Structure S1 (red) and S2 (blue) volume change (left) and energy change (right) vs Allegro model validation weighted sum error. Volume changes are given in $\mathring{A}^3$, energy changes are given in meV, and weighted sum values are unitless. In these plots, the data is given by the unfilled markers and the linear fit to the data is given by the lines.

We see that there is indeed an observable trend between both the validation error and the change in volume as well as the validation and the change in energy. This supports the assumption that the validation error is a good measure of interpolative capability of the model.

5.2.5 Molecular Dynamics and Conductivity Estimation

We now look at the conductivity results for the five small combined models that passed the 900K screening test, CS2 - CS6. Each of these potential models were used in LAMMPS to run simulations at 350K, 450K, and 550K. Ten simulations with different initial velocity distributions were ran at each temperature for 2.8 ns of simulation time. For each potential model, we can analyze the interval functions which are functions of the time interval which is being averaged over, τ, as described in the preprint in Appendix C.

There are three types of interval functions: $\overline{\Lambda}^{\chi}_{\phi,v}(\tau)$, which is averaging over a block intervals of length τ for a single Allegro model ϕ and single velocity distribution v, $\overline{\overline{\Lambda}}^{\chi}_{\phi}(\tau)$, which additionally averages over the initial velocity distributions of the Allegro model ϕ, and $\overline{\overline{\overline{\Lambda}}}^{\chi}(\tau)$, which additionally averages over all five similar Allegro models. Here χ is either "Tr" or "Full", denoting which approach we are using when defining the interval functions.

The full definitions of the interval functions and their errors are given in Section III-B of the preprint in Appendix C. Below we see select results of $\overline{\Lambda}^{\chi}_{\phi,v}(\tau)$ and $\overline{\overline{\Lambda}}^{\chi}_{\phi}(\tau)$ from Allegro model CS2 as well as the average over all five models, $\overline{\overline{\overline{\Lambda}}}^{\chi}(\tau)$ for structure S1. These results are given in Figures 5.18 and 5.19. Similar plots for the other models and for structure S2 are given in the preprint in Appendix C.

Here we see that the majority of the error associated with averaging comes from variability of the velocity distribution, that the error associated with Allegro model variability is much smaller, but not trivial, and that the block averaging error is almost imperceptible. This finding suggests that the Allegro models are in fairly good agreement, that the block averaging is well controlled, and that we need to consider many more trajectories with different initial velocity distributions in future works.

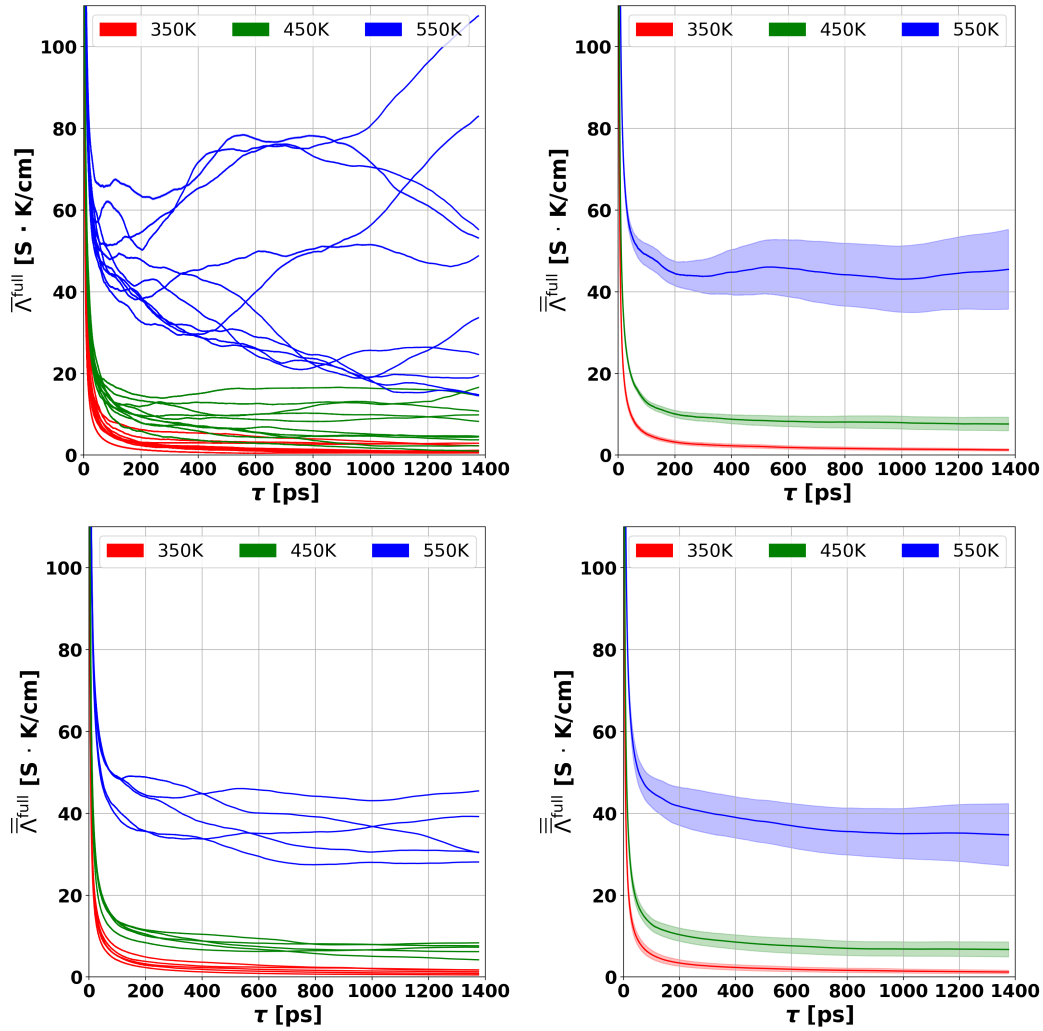


Figure 5.18: Figure showing the interval functions $\overline{\Lambda}^{\text{full}}_{\text{CS2},v}(\tau)$ with error shading (top left), $\overline{\overline{\Lambda}}^{\text{full}}_{\text{CS2}}(\tau)$ with error shading (top right), $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ without error shading (bottom left), and $\overline{\overline{\Lambda}}^{\text{full}}(\tau)$ with error shading (bottom right) for structure S1. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve.

We also see that in general the tracer conductivity is much better controlled than the full conductivity, and that for both approaches, at 550K the error and interval function behavior at long τ is still not well controlled. From these interval functions, we can perform a linear regression to estimate the conductivity, and by extension, the Haven ratio. These results are provided in Tables VII and VIII of the preprint in

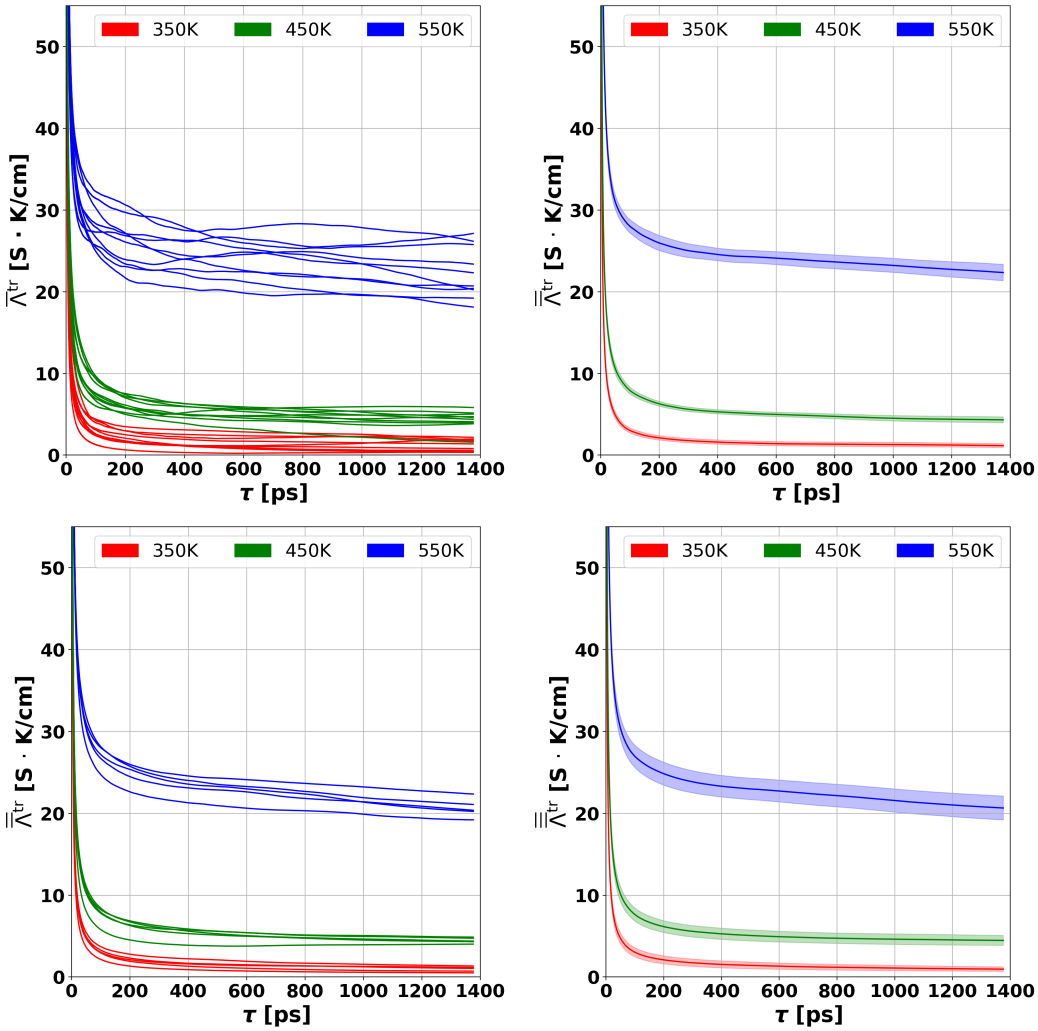


Figure 5.19: Figure showing the interval functions $\overline{\Lambda}^{\mathrm{tr}}_{\mathrm{CS2},v}(\tau)$ with error shading (top left), $\overline{\overline{\Lambda}}^{\mathrm{tr}}_{\mathrm{CS2}}(\tau)$ with error shading (top right), $\overline{\overline{\Lambda}}^{\mathrm{tr}}_{\Phi}(\tau)$ without error shading (bottom left), and $\overline{\overline{\overline{\Lambda}}}^{\mathrm{tr}}(\tau)$ with error shading (bottom right) for structure S1. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve.

Appendix C.

From these results we see that the Haven ratio is less than 1.0 and the material seems to have prominent ionic correlation. The conductivity estimates presented suggest that structure S1 has higher conductivity than structure S2, roughly 4.13 compared to 3.52 mS/cm at temperatures around 350K. We can additionally trans-

form our estimates of conductivity and temperature into the correct functional form to construct an Arrhenius plot. For these Arrhenius plots, we can then estimate the activation energy of the material based on the slope of the Arrhenius plots. The Arrhenius plots are shown below in Figures 5.20 and 5.21. The activation energies are shown Table IX in the preprint in Appendix C.

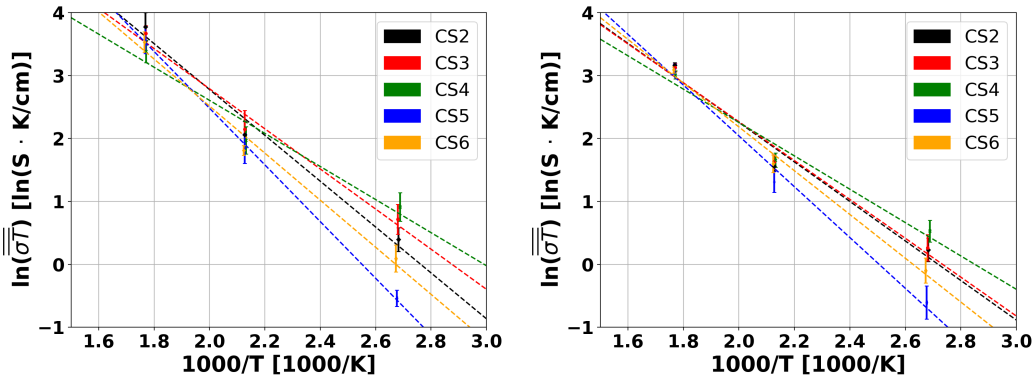


Figure 5.20: Arrhenius plots of full (left) and tracer (right) conductivity results for each of the Allegro models CS2 - CS6 for structure S1.

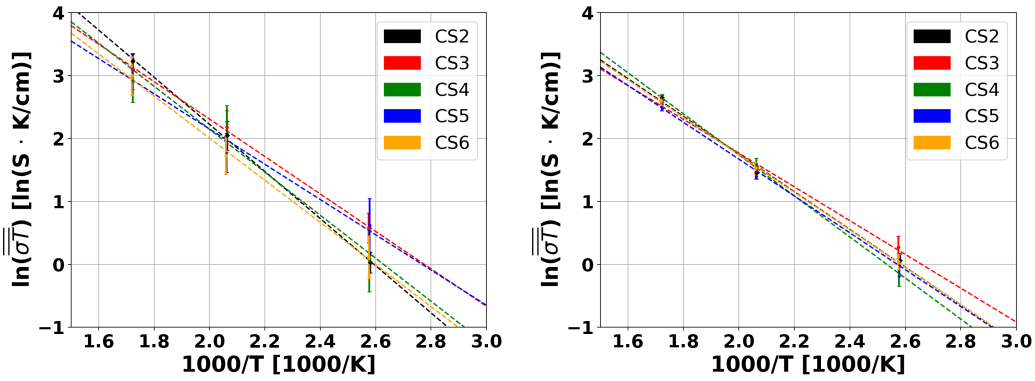


Figure 5.21: Arrhenius plots of full (left) and tracer (right) conductivity results for each of the Allegro models CS2 - CS6 for structure S2.

We note that the slope of the Arrhenius plots for the individual models which are constructed from the double averaged interval functions are quite different, which impacts our confidence in the estimations of the activation energy. This may in part

be due to the fact that we are fitting to only three temperatures here, which is an insufficient number of datapoints for the fitting process in general. For future work, if we wish to estimate the activation energy with some degree of confidence, we will certainly need to include additional datapoints at different temperatures to ensure a better fit.

This study showcases that for this test material, the tracer conductivity is not sufficient due to significant ionic correlation, and that by using the Allegro potentials we can reach the much longer simulation times needed to allow for sufficient averaging to estimate the full conductivity. We also see that although the block averaging is controlling one source of error, and although the error associated with the Allegro training process is not trivial, a large portion of the error is associated with the choice of velocity initialization of the particles. In future works, we expect to take advantage of this new information by sampling a much larger set of initial velocity distributions.

5.2.6 Comparison of Single vs. Combined Structural Training

Here we quickly note a few results from the single structure models SS1 and SS2. Performing an analysis of the interval functions, we see that they exhibit similar behavior as the combined models, as shown in Figures 5.22 and 5.23 below for the full conductivity approach. We note that for these models there are no triply averaged interval functions as there is only one model for each structure.

We can similarly estimate conductivities and Haven ratios, seen in Table X in the preprint in Appendix C.

Finally, we can also generate Arrhenius plots, which we compare to the combined training model average from the previous section below in Figure 5.24. Again, we can use these results to estimate activation energies, which are similarly of low confidence,

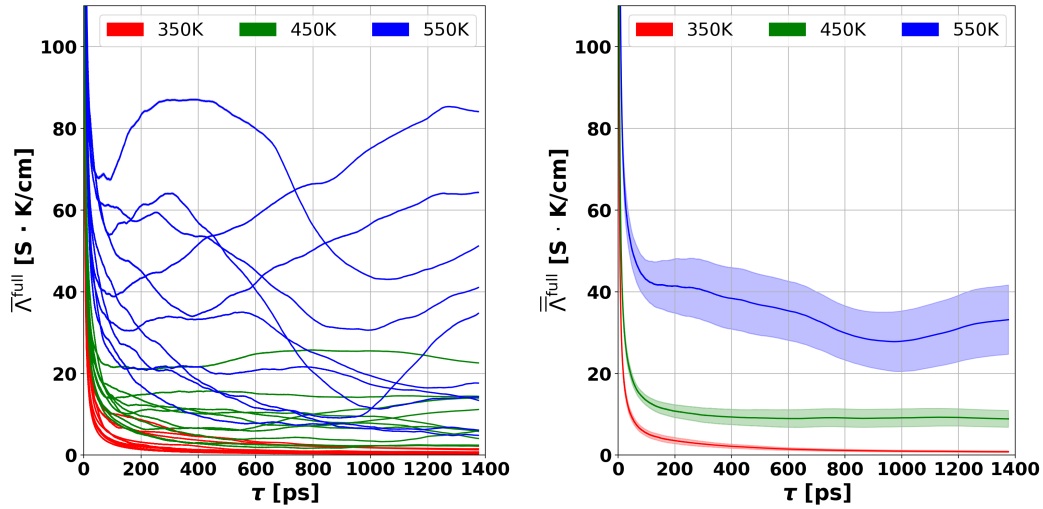


Figure 5.22: Figure showing the interval functions $\overline{\Lambda}^{\text{full}}_{\text{SS1},v}(\tau)$ with error shading (left), and $\overline{\overline{\Lambda}}^{\text{full}}_{\text{SS1}}(\tau)$ with error shading (right) for structure S1. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve.

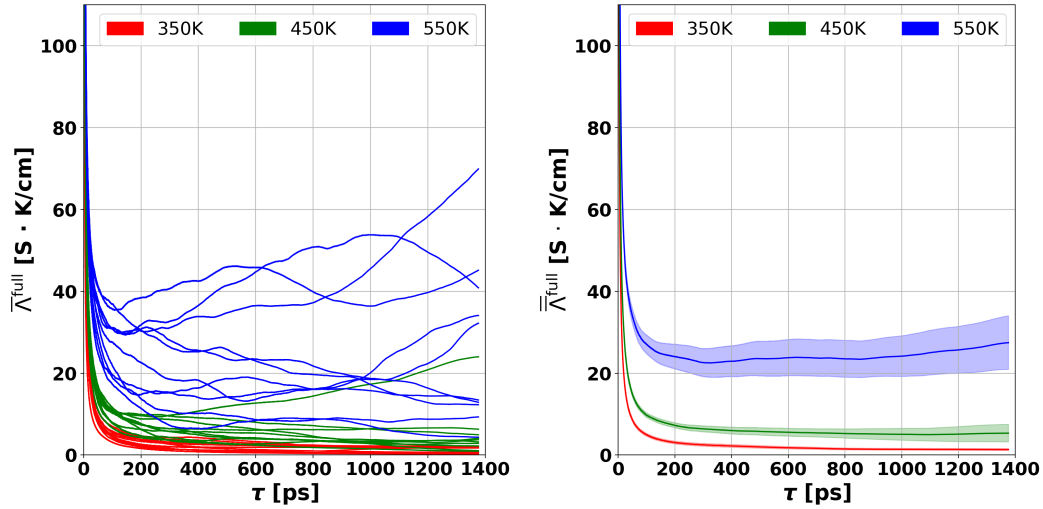


Figure 5.23: Figure showing the interval functions $\overline{\Lambda}^{\text{full}}_{\text{SS2},v}(\tau)$ with error shading (left), and $\overline{\overline{\Lambda}}^{\text{full}}_{\text{SS2}}(\tau)$ with error shading (right) for structure S2. The interval function curve is given by the darker color, and the error is given by lighter colored shading surrounding the curve.

and are given in Table XI of the preprint in Appendix C.

From these results we see that the errors are all quite similar to the combined model

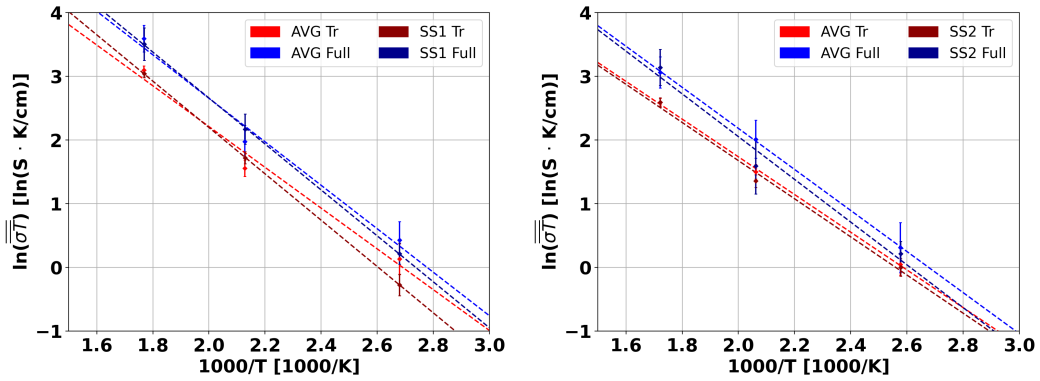


Figure 5.24: Allegro model SS1 (left) and SS2 (right) average Arrhenius plots of full and tracer conductivity results for structures S1 and S2 respectively. Combined CS2 - CS6 model average results are included for comparison. Tracer conductivity estimates are given in red and full conductivity estimates are given in blue.

training and that the estimates of the conductivity, Haven ratio, and activation energy of both structures are in agreement with the combined model results, with the difference being within the margin of error. This suggests that the combined model training is not impacting the predictive capability for a particular structure negatively in this case, and that, given this preliminary evidence, in future work it may be a good idea to train a single model with multiple structures.

5.3 Estimating the ionic conductivity of select (thio)boracite solid electrolyte materials using the Allegro machine learned interatomic potential to extend first principles calculations

This section serves as an outline of the project we are currently working on, which is still in the early stages of development and will continue well into the future, with the goal of becoming a third publication. With first principles calculations alone, we were unable to quantitatively assess the performance of the boracite and thioboracite materials with regards to their lithium ion conductivity. The results

from our investigation into using Allegro to extend first principles calculations has encouraged us to apply this approach to these materials, where it now seems possible to run the long molecular dynamics simulations necessary in order to determine their ionic conductivity.

5.3.1 Project Scope and Objectives

Ideally, I would like to attempt to utilize Allegro to study all eight boracite and thioboracite materials that were discussed in section 5.1. Given the time constraints, I have decided to focus my attention on a smaller set of materials. The materials with fewer atomic species are less complex, therefore I am proposing to focus on the materials without aluminum: $Li_4B_7O_{12}Cl$, $Li_6B_7O_{13}Cl$, $Li_4B_7S_{12}Cl$, and $Li_6B_7S_{13}Cl$. There are several questions I wish to answer by completing this project:

- Can we utilize Allegro to extend short first principles molecular dynamics simulations for the boracite and thioboracite materials and maintain the accuracy of first principles?
- Can we overcome the noise associated with calculating the full conductivity for these materials, allowing us to more accurately estimate the conductivity?
- How important are the ionic correlations for calculating the conductivity in these materials? Is the tracer approximation valid for these materials?
- Can we extend the concept of "training diversity" to include multiple different materials when training an Allegro model rather than just multiple different structures of a single material?
- Can we increase our confidence in the activation energy estimates derived from the Arrhenius plots?

- How do the estimates of conductivity and activation energies compare to experimental data?

5.3.2 Computational Details and Timeline

The computational steps are listed in order, along with timing estimates and current status at the time of this publication, in Table 5.3 below.

Table 5.3: Computational details for the current project.

Task	Est. Time	Status
Relax equilibrium structures for running FPMD	3 days	Complete
Run FPMD on equilibrium structures	3 weeks	Complete
Construct training datasets	1 week	Complete
Run Allegro training	2 weeks	Not Started
Test Allegro models	1 week	Not started
Analyze results and write paper	1 month	Not started

The expectation is that Allegro training will start the week of August 9, 2026 and will be completed by the week of August 23, 2026. The testing of the trained models and analysis should then be completed by the week of August 30, 2026, at which point the paper will need to be written and submitted for publication. I fully intend to continue my collaborative efforts with Dr. Holzwarth throughout the remainder of this project.

5.4 A Study of Silicon Doped Argyrodite in Collaboration with Argonne National Laboratory

5.4.1 Determining the Optimal Substitution Sites for ZnO

Our collaborative efforts focused on modeling possible structures of the ZnO doped lithium argyrodite material being experimentally studied by Dr. Zachary Hood and Udo Eze. Some background information regarding this material is discussed in section

1.4. In our modeling efforts we realized that there was some ambiguity regarding the ground-state structure for the pure lithium argyrodite material, prompting us to consider two possible starting structures: an $F\bar{4}3m$ structure with fully occupied $24g$ lithium sites as well as an $F\bar{4}3m$ structure with 50% occupied $48h$ lithium sites. After optimizing these cells, we first attempted to find the oxygen substitution site based on symmetry considerations, where oxygen can either replace the lone sulfur atom in the material or replace a sulfur within a PS_4 unit. We found the energetically favorable substitution site for oxygen to be one of the four sulfur sites in the PS_4 unit, which was in agreement with previous research done on this material [87].

We then began to attempt to determine where the zinc and oxygen should be placed for the ZnO substitution. While an exhaustive search was not performed, utilizing symmetry considerations in conjunction with the known oxygen substitution site, we were able to systematically search a fairly large subset of possible structures with specific ZnO placement. This was done by considering all possible sites where zinc could replace lithium in the pure material and then additionally removing the nearest secondary lithium to maintain charge neutrality within the cell, resulting in 23 possible configurations. The configuration with the lowest energy after relaxation was then further tested to determine where to place oxygen, resulting in 20 additional possible configurations. After a final set of relaxations, the lowest energy configuration with the full ZnO substitution was determined, where the cell had distorted, with oxygen once again replacing a sulfur in a PS_4 unit and zinc replacing a nearby lithium such that the zinc and oxygen were withing the bonding distance of one another. A candidate pure as well as ZnO substituted material from this structural search was constructed as a cubic cell in order to maintain consistency with the experimental data. Additionally, a third material was considered by taking the ZnO doped stochiometry and increasing the amount of chlorine per unit cell, which requires

Table 5.4: Table detailing the properties of the three candidate structures for the lithium argyrodite materials investigated in this study, $Li_{24}P_4S_{20}Cl_4$, $Li_{22}P_4S_{19}Cl_4ZnO$, and $Li_{21}P_4S_{18}Cl_5ZnO$. Since the structures are cubic, there is only one lattice parameter to define, which is a, given in Angstroms. The precursor materials which serve as potential decomposition products are listed, and the formation energy is given with respect to these materials, indicating stability with respect to decomposition.

Material	a (Å)	ΔE_F (eV)	Precursors
$Li_{24}P_4S_{20}Cl_4$	9.8	-2.3	$10Li_2S + 4LiCl + 2P_2S_5$
$Li_{22}P_4S_{19}Cl_4ZnO$	9.7	-3.6	$9Li_2S + 4LiCl + 2P_2S_5 + ZnO$
$Li_{21}P_4S_{18}Cl_5ZnO$	9.7	-3.4	$8Li_2S + 5LiCl + 2P_2S_5 + ZnO$

a removal of one unit of Li_2S and addition of one unit of $LiCl$ in the set of precursor materials. The resultant three cubic structures - $Li_{24}P_4S_P20Cl_4$, $Li_{22}P_4S_P19Cl_4ZnO$, and $Li_{21}P_4S_P18Cl_5ZnO$ - were relaxed in quantum espresso with fixed lattice vectors to maintain their cubic symmetry. The results comparing the structural properties is given below in Table 5.4:

We see that in all three cases, the material is stable with respect to its decomposition products. We also see that both ZnO materials are energetically favorable compared to the pure material. The formation energy can be defined ,for example, as seen in the equation below, where $U_{\mathrm{SL}}(ABC)$ indicates static lattice energy of the compound ABC:

$$\Delta E_F(Li_{24}P_4S_{20}Cl_4) = U_{\mathrm{SL}}(Li_{24}P_4S_{20}Cl_4) - 10U_{\mathrm{SL}}(Li_2S) - 4U_{\mathrm{SL}}(LiCl) - 2U_{\mathrm{SL}}(P_2S_5) \tag{5.1}$$

Similar equations can be defined for the other two compositions. The structures can be visualized in VESTA, which is shown in Figure 5.25.

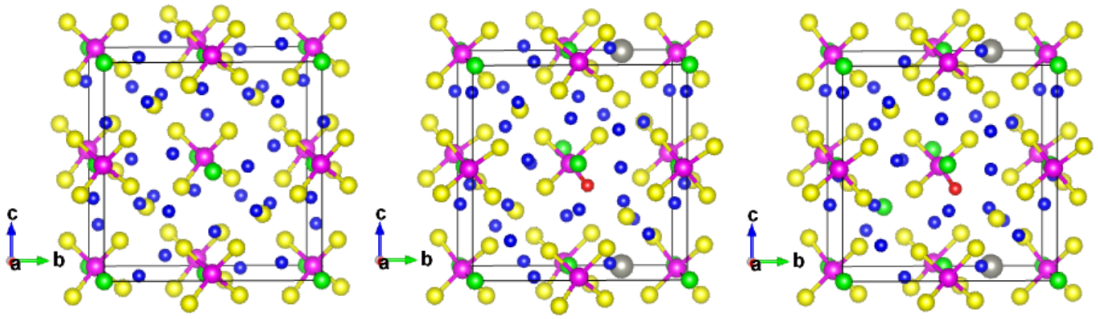


Figure 5.25: Idealized candidate cubic structures for $Li_{24}P_4S_{20}Cl_4$ (left), $Li_{22}P_4S_{19}Cl_4ZnO$ (center), and $Li_{21}P_4S_{18}Cl_5ZnO$ (right). Here blue, magenta, grey, yellow, red, and green spheres represent lithium, phosphorous, silicon, sulfur, oxygen, and chlorine.

5.4.2 Computed X-Ray Diffraction Patterns

In addition to visualizing the structures, we can also compute x-ray diffraction results as a function of the incident x-ray angle θ. In Figure 5.26 below, we see the computed x-ray results compared to experimental data.

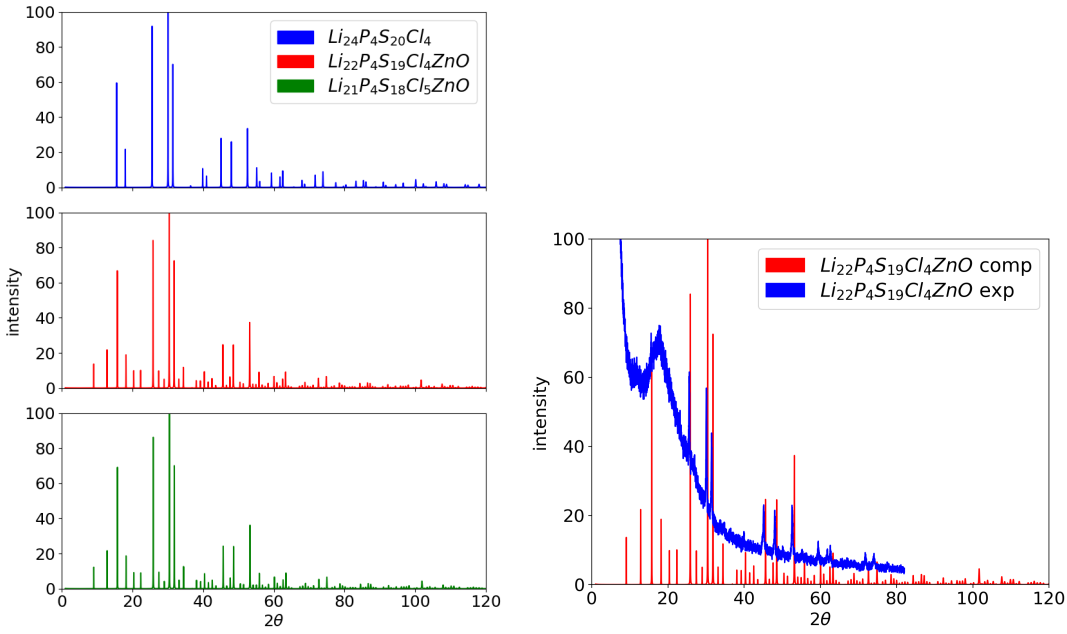


Figure 5.26: X-ray results for all three idealized cubic structures determined computationally (left) and the idealized $Li_{22}P_4S_{19}Cl_4ZnO$ cubic structure's computed x-ray data compared to its experimental counterpart (right).

From the computed x-ray data, we see that the patterns across the three materials are quite similar overall, but have unique characteristics to differentiate them from one another, such as missing peaks or different peak magnitudes. We also see that the computed $Li_{22}P_4S_{19}Cl_4ZnO$ results match very nicely to the experimental x-ray data. We had began to investigate the performance of the three idealized structures by running molecular dynamics, but unfortunately these results were too preliminary to consider seriously and are not included here. Since the project was dropped due to circumstances outside of our control, further investigation of these materials was not continued. The materials are still of interest and research involving these materials is continuing, as evidenced by a recent paper by Jang et al. in 2023 [88].

Appendix A: Selected Practical Computational Details and Example Code

A.1 Pytorch Neural Network Example

This section includes the Python code for the simple example in Section 3.3.1. This code was written based on the pytorch framework [89] and also utilizes the matplotlib [90] library. We note that this is a simplified example for illustrative purposes and is not a fully featured FFNN with realistic training and validation considerations. For instance, the code only produces per batch errors for training and testing, which is not reproducible due to the shuffle settings set to true in the dataloader. Additionally, the datasets must be constructed to be integer multiples of the batch size to avoid losing the last few training/validation examples that don't fit in a batch, due to drop_last being set to true in the dataloader.

neural network training code

```
# import necessary python packages
import torch
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
import numpy as np
import matplotlib.pyplot as plt
import sys

# prepare output log
logfile = open("train.log", "w")

# define our NN class
class NeuralNetwork(torch.nn.Module):
        def __init__(self, num_inputs, num_outputs, width):
                super().__init__()
```

```
                self.layers = torch.nn.Sequential(

                        # layer 1
                        torch.nn.Linear(num_inputs, width),
                        torch.nn.SiLU(),

                        # layer 2
                        torch.nn.Linear(width, width),
                        torch.nn.SiLU(),

                        # output layer
                        torch.nn.Linear(width, num_outputs),
                )

        def forward(self, x):
                return self.layers(x)

# define our network dimensions
num_inputs = 2    # 2 atoms with q_1*q_2 and r_12 data
num_outputs = 1   # output a single scalar representing the energy
width = 1024        # 1024 neurons per layer

# specify random seed for model weight initialization
torch.manual_seed(123456)

# instantiate a model with weights initialized based on random seed
model = NeuralNetwork(num_inputs, num_outputs, width)
print("constructing␣network...")
print(model)
print("constructing␣network...", file=logfile)
print(model, file=logfile)

# calculate the total number of weights in the network
# check total number of weights in the network
num_weights = sum(
        p.numel() for p in model.parameters() if p.requires_grad
)
print(f"total␣number␣of␣weights␣in␣the␣network:␣{num_weights}")
print(f"total␣number␣of␣weights␣in␣the␣network:␣{num_weights}", file
    =logfile)

# read in the csv file from the first command line argument,
    skipping the first row
# first row should be the header with column labels
input_dataset = np.loadtxt(sys.argv[1], delimiter = ',', skiprows =
    1, comments = "#", dtype = np.float32)

# slice the np array to separate the inputs and outputs
# row of data is a training example of the form: q_1*q_2, r_12, E
Xtrain_np = input_dataset[:,:2]
Ytrain_np = input_dataset[:,2:3]
```

```
# apply normalization to the training data to force inputs/outputs
    to be in range [-1,1]
X_min = Xtrain_np.min()
X_max = Xtrain_np.max()
for row_idx, row in enumerate(Xtrain_np):
        for col_idx, elem in enumerate(row):
                Xtrain_np[row_idx][col_idx] = elem / max([abs(X_min)
                    ,X_max])
Y_min = Ytrain_np.min()
Y_max = Ytrain_np.max()
for row_idx, row in enumerate(Ytrain_np):
        for col_idx, elem in enumerate(row):
                Ytrain_np[row_idx][col_idx] = elem / max([abs(Y_min)
                    ,Y_max])
print("normalization␣of␣training␣data:␣X_norm␣=␣X␣/␣␣max([abs(X_min)
    ,X_max])")
print(f"normalization␣of␣inputs␣X␣␣--␣min:␣{X_min}␣␣␣max:␣{X_max}")
print(f"normalization␣of␣outputs␣y␣--␣min:␣{Y_min}␣␣␣max:␣{Y_max}")
print("normalization␣of␣training␣data:␣X_norm␣=␣X␣/␣␣max([abs(X_min)
    ,X_max])", file=logfile)
print(f"normalization␣of␣inputs␣X␣␣--␣min:␣{X_min}␣␣␣max:␣{X_max}",
    file=logfile)
print(f"normalization␣of␣outputs␣y␣--␣min:␣{Y_min}␣␣␣max:␣{Y_max}",
    file=logfile)
print(Xtrain_np)
print(Ytrain_np)
print(f"X␣value␣scaling␣factor:␣{max([abs(X_min),X_max])}")
print(f"Y␣value␣scaling␣factor:␣{max([abs(Y_min),Y_max])}")
print(f"X␣value␣scaling␣factor:␣{max([abs(X_min),X_max])}", file =
    logfile)
print(f"Y␣value␣scaling␣factor:␣{max([abs(Y_min),Y_max])}", file =
    logfile)

# convert them to pytorch tensor objects
Xtrain_tt = torch.from_numpy(Xtrain_np)
Ytrain_tt = torch.from_numpy(Ytrain_np)
print(f"Xtrain_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Xtrain_tt.shape
    }␣and␣type:␣{Xtrain_tt.dtype}")
print(f"Ytrain_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Ytrain_tt.shape
    }␣and␣type:␣{Ytrain_tt.dtype}")
print(f"Xtrain_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Xtrain_tt.shape
    }␣and␣type:␣{Xtrain_tt.dtype}", file=logfile)
print(f"Ytrain_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Ytrain_tt.shape
    }␣and␣type:␣{Ytrain_tt.dtype}", file=logfile)

# define a dataset class to use in conjunction with DataLoader to
    load our dataset
class MyDataset(Dataset):
        def __init__(self, X, y):
                self.features = X
```

```
                self.outputs = y

        def __getitem__(self,index):
                one_x = self.features[index]
                one_y = self.outputs[index]
                return one_x, one_y

        def __len__(self):
                return self.outputs.shape[0]

# create a MyDataset object with our training data
train_ds = MyDataset(Xtrain_tt, Ytrain_tt)

# specify random seed for shuffling the dataset upon loading
torch.manual_seed(123456)

# specify the batch size
b_size = 3

# create a DataLoader object from our MyDataset object
train_loader = DataLoader(
        dataset = train_ds,
        batch_size = b_size,
        shuffle = True,
        num_workers = 0,
        drop_last = True
)

# define settings for training
num_epochs = 500
learning_rate = 1e-3

# define the optimizer to be used to train - using SGD here
optimizer = torch.optim.SGD(model.parameters(), lr = learning_rate)

# define the loss function to quantify the error - using L2 loss
   here
loss_fn = torch.nn.MSELoss()

# define a training function
def trainNN(train_loader, model, loss_fn, optimizer, epoch):
        model.train()
        for batch, (X, y) in enumerate(train_loader):
                # get the model's prediction
                pred = model(X)

                # calculate the loss
                loss = loss_fn(pred, y)

                # optimize the weights
                optimizer.zero_grad()
```

```
                loss.backward()
                optimizer.step()

                print(f"Epoch: {epoch + 1}    |    Batch: {batch + 
                    1}/{len(train_loader)}    |    Loss: {loss:.6f}")
                print(f"Epoch: {epoch + 1}    |    Batch: {batch + 
                    1}/{len(train_loader)}    |    Loss: {loss:.6f}",
                    file=logfile)
        print("-----------------------------------")
        print("-----------------------------------", file=logfile)

# train the network
for epoch in range(num_epochs):
        trainNN(train_loader, model, loss_fn, optimizer, epoch)
print("training complete")
print("training complete", file=logfile)

# save the trained model to a .pth file
torch.save(model.state_dict(), "trained_model.pth")
print("saved model to trained_model.pth")
print("saved model to trained_model.pth", file=logfile)

# close the output log file
logfile.close()
```

neural network testing code

```
# import necessary python packages
import torch
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
import numpy as np
import matplotlib.pyplot as plt
import sys, math

# prepare output log
logfile = open("test.log", "w")

# define our NN class
class NeuralNetwork(torch.nn.Module):
        def __init__(self, num_inputs, num_outputs, width):
                super().__init__()

                self.layers = torch.nn.Sequential(

                        # layer 1
                        torch.nn.Linear(num_inputs, width),
                        torch.nn.SiLU(),
```

```
                        # layer 2
                        torch.nn.Linear(width, width),
                        torch.nn.SiLU(),

                        # output layer
                        torch.nn.Linear(width, num_outputs),
                )

        def forward(self, x):
                return self.layers(x)

# define our network dimensions
num_inputs = 2    # 2 atoms with q_1*q_2 and r_12 data
num_outputs = 1   # output a single scalar representing the energy
    of the pair
width = 1024        # 1024 neurons per layer

# instantiate a model
model = NeuralNetwork(num_inputs, num_outputs, width)
print("constructing␣network...")
print(model)
print("constructing␣network...", file=logfile)
print(model, file=logfile)

# read in the saved model to get the trained weights
model.load_state_dict(torch.load("trained_model.pth", weights_only=
    True))
print("loaded␣model␣from␣trained_model.pth")
print("loaded␣model␣from␣trained_model.pth", file=logfile)

# calculate the total number of weights in the network
# check total number of weights in the network
num_weights = sum(
        p.numel() for p in model.parameters() if p.requires_grad
)
print(f"total␣number␣of␣weights␣in␣the␣network:␣{num_weights}")
print(f"total␣number␣of␣weights␣in␣the␣network:␣{num_weights}", file
    =logfile)

# read in the csv file from the first command line argument,
    skipping the first row
# first row should be the header with column labels
input_dataset = np.loadtxt(sys.argv[1], delimiter = ',', skiprows =
    1, comments = "#", dtype = np.float32)

# slice the np array to separate the inputs and outputs
# row of data is a training example of the form: q_1*q_2, r_12, E
q1q2 = input_dataset[0,0]
Xtest_np = input_dataset[:,:2]
Ytest_np = input_dataset[:,2:3]
```

```
# apply normalization to the testing data to force inputs/outputs to
    be in range [-1,1] (note these must be manually defined from
    training output)
x_scaling_factor = 3.9990854263305664
y_scaling_factor = 1.00112783908844

for row_idx, row in enumerate(Xtest_np):
        for col_idx, elem in enumerate(row):
                Xtest_np[row_idx][col_idx] = elem / x_scaling_factor

for row_idx, row in enumerate(Ytest_np):
        for col_idx, elem in enumerate(row):
                Ytest_np[row_idx][col_idx] = elem / y_scaling_factor

print("normalization␣of␣testing␣data␣complete")
print("normalization␣of␣testing␣data␣complete", file = logfile)

# convert them to pytorch tensor objects
Xtest_tt = torch.from_numpy(Xtest_np)
Ytest_tt = torch.from_numpy(Ytest_np)
print(f"Xtest_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Xtest_tt.shape}␣
    and␣type:␣{Xtest_tt.dtype}")
print(f"Ytest_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Ytest_tt.shape}␣
    and␣type:␣{Ytest_tt.dtype}")
print(f"Xtest_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Xtest_tt.shape}␣
    and␣type:␣{Xtest_tt.dtype}", file=logfile)
print(f"Ytest_tt␣loaded␣as␣tensor␣object␣of␣shape:␣{Ytest_tt.shape}␣
    and␣type:␣{Ytest_tt.dtype}", file=logfile)

# define a dataset class to use in conjunction with DataLoader to
    load our dataset
class MyDataset(Dataset):
        def __init__(self, X, y):
                self.features = X
                self.outputs = y

        def __getitem__(self,index):
                one_x = self.features[index]
                one_y = self.outputs[index]
                return one_x, one_y

        def __len__(self):
                return self.outputs.shape[0]

# create a MyDataset object with our training data
test_ds = MyDataset(Xtest_tt, Ytest_tt)

# specify random seed for shuffling the dataset upon loading
torch.manual_seed(123456)
```

```
# specify the batch size
b_size = 2

# create a DataLoader object from our MyDataset object
test_loader = DataLoader(
        dataset = test_ds,
        batch_size = b_size,
        shuffle = True,
        num_workers = 0,
        drop_last = True
)

# define the loss function to quantify the error - using L2 loss for
    this example
loss_fn = torch.nn.MSELoss()

# define a testing function
def testNN(test_loader, model, loss_fn):
        model.eval()
        with torch.no_grad():
                for batch, (X, y) in enumerate(test_loader):
                        # get the model's prediction
                        pred = model(X)

                        # calculate the loss
                        loss = loss_fn(pred, y)

                        print(f"Testing:␣␣␣|␣␣␣Batch:␣{batch␣+␣1}/{
                            len(test_loader)}␣␣␣|␣␣␣Loss:␣{loss:.6f}"
                            )
                        print(f"Testing:␣␣␣|␣␣␣Batch:␣{batch␣+␣1}/{
                            len(test_loader)}␣␣␣|␣␣␣Loss:␣{loss:.6f}"
                            , file=logfile)
        print("-----------------------------------")
        print("-----------------------------------", file=logfile)

# test the network
testNN(test_loader, model, loss_fn)
print("testing␣complete")
print("testing␣complete", file=logfile)

# check if command line argument provided to plot
# --plot=true
create_plot = False
for i in range(len(sys.argv)):
        if sys.argv[i] == "--plot=true":
                create_plot = True

# plot if specified
if create_plot:
        # calculate the predictions for all inputs from the test
```

```
            dataset
        model.eval()
        with torch.no_grad():
                predictions = model(Xtest_tt)

        # get the input displacements
        distances = []
        for i in range(len(Xtest_tt)):
                distances.append(Xtest_tt[i][1].item())

        # convert predictions to list of numbers to plot
        energies = []
        for i in range(len(predictions)):
                energies.append(predictions[i][0].item())

        # don't forget to convert to un-normalized values
        for i in range(len(energies)):
                energies[i] *= y_scaling_factor
                distances[i] *= x_scaling_factor

        # generate true data
        true_x = np.arange(min(distances), max(distances), 0.01)
        true_y = q1q2/true_x

        # plot the data
        plt.title("Plot of test data")
        plt.ylabel("U(r)")
        plt.xlabel("r")
        plt.plot(distances, energies, label="test data", color = "
            purple", marker = "x", linestyle = "None")
        plt.plot(true_x, true_y, label="true fn", color = "green",
            marker = "None", linestyle = "dashed")
        plt.legend(loc = "upper right")
        plt.savefig("testdata.png")

        print("saved scatter plot of test data energy vs distance:
            testdata.png")
        print("saved scatter plot of test data energy vs distance:
            testdata.png", file=logfile)

# close the output log file
logfile.close()
```

training dataset creation code

```
# script to generate datasets
import numpy as np
```

```
# training data
x_range = [1.0,4.0]
noise_level = 0.01
rand_seed = 29456812
num_examples = 2048
q1 = 1.0
q2 = 1.0
q1q2 = q1 * q2

# setup rng
rng = np.random.default_rng(seed=rand_seed)

# generate noisy training data
x_data = rng.uniform(low=x_range[0], high=x_range[1], size=
    num_examples).tolist()
y_data = [(q1q2/r) + (noise_level * rng.uniform(low=-1.0, high=1.0))
     for r in x_data]

# save the training data
trainOut = open("train_dataset.csv", "w")
print("q1*q2,r12,U", file = trainOut)
for i in range(num_examples):
        print(f"{q1q2:.12f},{x_data[i]:.12f},{y_data[i]:.12f}", file
            = trainOut)
trainOut.close()
```

A.2 Close-Pair Dataset Construction

This section contains the code for the close-pair dataset construction method discussed in Section 5.2. The code finds all pairs of type $X - Y$ and $Y - X$ where X is the first atom type listed in the code (in our case, lithium) and Y is any other atomic type in the material. It then proceeds to iteratively move the first atom in the pair towards the second in a direct line of motion by a defined shift value, until the pair has reached the minimum pair distance allowed by the PAW orbital radii of the constituent atoms. In the process, it checks each shifted location and discards any shifted positions where atom has become too close to any other atom in the cell. This protocol is not exhaustive, but seems to work well. One could imagine improving this by considering shifting by angular increments as well as radial increments, or

by considering more types of pairs to include in the shifting procedure besides just those containing the first atom type. In this case, it made sense to limit the pair selection in the way because the first atom type is lithium, which is the mobile ion. The remaining atom types are framework atoms, which should not be moving much nor should they be incidentally coming to close in the course of the MD simulations.

close-pair construction code

```
# this script reads in an input structure and creates all possible
    pairs and their local neighborhoods

import fileinput, os, sys, math

# function to check if atom is too close to other atoms
def isPositionValid(atomID, newPosition, pairList, minDistList):
    # atomID = Li4 for instance
    # pairList is the list of all pairs
    # minDistList is the list of all possible pair types and their
        closest distances
    for line in pairList:
        # look for pairs where the first atom is the one we are
            moving
        if line[0].split('-')[0] == atomID:
            # loop through the pair types to get the right one
            pair_type_found = False
            pair_type_ind = 0
            while not pair_type_found:
                if line[1] == minDistList[pair_type_ind][0]:
                    pair_type_found = True
                else:
                    pair_type_ind += 1
            # found the pair type index... check if it's too close
            newDist = math.sqrt((newPosition[0] - line[5])**2.0 + (
                newPosition[1] - line[6])**2.0 + (newPosition[2] -
                line[7])**2.0)
            if newDist < minDistList[pair_type_ind][1]:
                return False
    # if we get through the whole list, we're good to go
    return True

# input variables
input_xyz_path = sys.argv[1]
input_qe_template_path = sys.argv[2]
r_cut = float(sys.argv[3])
shift_value = float(sys.argv[4])
```

```
paw_radii = [float(i) for i in sys.argv[5].split(",")]

# read in the qe template
qe_out_lines = []
for line in fileinput.input(files=input_qe_template_path):
    qe_out_lines.append(line.rstrip('\n'))

# get the target line to insert atomic positions and cell parameters
target_qe_line = "TARGET_LINE_ATOMIC_POS_AND_CELL_PARAMS"
target_qe_line_ind = 0
for i in range(len(qe_out_lines)):
    if qe_out_lines[i] == target_qe_line:
        target_qe_line_ind = i

# read in the structure
input_lines = []
for line in fileinput.input(files=input_xyz_path):
    input_lines.append(line.rstrip('\n').split())

# get the number of atoms
num_atoms = int(input_lines[0][0])

# get the cell dimensions
cell_a = []
cell_b = []
cell_c = []
cell_a.append(float(input_lines[1][0].lstrip("Lattice=\"")))
cell_a.append(float(input_lines[1][1]))
cell_a.append(float(input_lines[1][2]))
cell_b.append(float(input_lines[1][3]))
cell_b.append(float(input_lines[1][4]))
cell_b.append(float(input_lines[1][5]))
cell_c.append(float(input_lines[1][6]))
cell_c.append(float(input_lines[1][7]))
cell_c.append(float(input_lines[1][8].rstrip("\"")))

# delete header (2 lines)
del input_lines[0]
del input_lines[0]

# convert data to floats
for line in input_lines:
    for i in range(1,len(line)):
        line[i] = float(line[i])

# add atom labels i.e. Li1, Li2, Si1, etc.
species = input_lines[0][0]
atom_num = 1
num_species = 1
species_labels = [species]
species_num_atoms = [0]
```

```
for line in input_lines:
    if line[0] == species:
        label = species + str(atom_num)
        line.insert(0,label)
        atom_num += 1
        species_num_atoms[-1] += 1
    else:
        num_species += 1
        species = line[0]
        species_labels.append(species)
        atom_num = 1
        label = species + str(atom_num)
        line.insert(0,label)
        atom_num += 1
        species_num_atoms.append(1)

# create the pairs based on r_cut
pairs = []

for i in range(num_atoms):
    for j in range(num_atoms):
        diffX = input_lines[j][2]-input_lines[i][2]
        diffY = input_lines[j][3]-input_lines[i][3]
        diffZ = input_lines[j][4]-input_lines[i][4]
        dist = math.sqrt((diffX**2.0) + (diffY**2.0) + (diffZ**2.0))
        if i != j and dist <= r_cut:
            pairs.append([str(input_lines[i][0]) + "-" + str(
                input_lines[j][0]), str(input_lines[i][1]) + "-" +
                str(input_lines[j][1]), input_lines[i][2],
                input_lines[i][3], input_lines[i][4], input_lines[j
                ][2], input_lines[j][3], input_lines[j][4], diffX,
                diffY, diffZ, dist])

# sort the list based on distance
pairs.sort(key=lambda x: x[-1])

outfile = open("pairs.csv", "w")
print("# pair_id\t\tpair_type\t\tr_1\t\tr_2\t\t(r_2-r_1)\t\t|r_2 - r_1|", file = outfile)
for line in pairs:
    print(line[0] + "," + line[1] + "," + str(line[2]) + "," + str(
        line[3]) + "," + str(line[4]) + "," + str(line[5]) + "," +
        str(line[6]) + "," + str(line[7]) + "," + str(line[8]) + ","
        + str(line[9]) + "," + str(line[10]) + "," + str(line[11]),
        file = outfile)
outfile.close()

# get the pair types
pair_types = []
for i in range(num_species):
    for j in range(num_species):
```

```
        pair_types.append(species_labels[i] + "-" + species_labels[j
            ])
num_pair_types = len(pair_types)

# group the pairs by type, i.e. Li-O, Li-P, ...
groups = []
for i in range(num_pair_types):
    groups.append([])

for line in pairs:
    for i in range(num_pair_types):
        if line[1] == pair_types[i]:
            groups[i].append(line)

outfile = open("groups.csv", "w")
outstr = "# "
for type in pair_types:
    outstr += type + "\t\t"
print(outstr, file=outfile)

for i in range(len(pairs)):
    outstr = ""
    for j in range(num_pair_types):
        if i < len(groups[j]):
            outstr += f"{groups[j][i][11]:4f}" + ","
        else:
            outstr += ","
    print(outstr, file = outfile)
outfile.close()

# get the minimum pair distances
min_pair_dists = []

# radii are assumed to be in Bohr need to convert to Angstroms
bohr2ang = 0.529177249

# loop over atom pair types
for i in range(num_species):
    for j in range(num_species):
        # calc min pair distance for ij pair converted to angstroms
        tmp_pair = species_labels[i] + "-" + species_labels[j]
        tmp_dist = bohr2ang*(paw_radii[i] + paw_radii[j])
        min_pair_dists.append([tmp_pair, tmp_dist])

# print the minimum pair distance results to a file
outfile = open("min_pair_dists.dat", "w")
print("min pair distances based on atomic PAW orbital radii not
   overlapping in Angstroms\n", file = outfile)
for i in range(len(min_pair_dists)):
    print(f"{min_pair_dists[i][0]}:   {min_pair_dists[i][1]}", file
        = outfile)
```

```
print("", file = outfile)
outfile.close()

# generated shifted structures for future scf calculations

# get the output lines...
output_lines = []
for i in range(num_atoms):
    outstr = str(input_lines[i][1]) + "␣␣␣␣␣" + str(input_lines[i
        ][2]) + "␣␣␣␣␣" + str(input_lines[i][3]) + "␣␣␣␣␣" + str(
        input_lines[i][4])
    output_lines.append(outstr)

# output info
num_shifted_structs = 0
out_info = []
# loop through all atoms of type 1, i.e. Li...
for i in range(species_num_atoms[0]):
    # loop through all pair types
    for j in range(num_species):
        # reset shift index
        k = 1
        # get the index for Li_i with the smallest pair distance of
            this type
        found_li = False
        min_pair_index = 0
        target_id = "Li" + str(i+1)
        while not found_li:
            # make sure we arent out of bounds
            if min_pair_index < len(groups[j]):
                if groups[j][min_pair_index][0].split('-')[0] ==
                    target_id:
                    found_li = True
                else:
                    min_pair_index += 1
            else:
                # out of bounds, this means we didn't find the
                    target in the list...
                break
        # if we found a target to shift in the list...
        if found_li:
            # loop through n shifts for each min pair to move it
                closer
            looping = True
        else:
            looping = False
            out_info.append([])
            out_info[-1].append("***␣struct-" + str(i) + "-" + str(j
                ) + "-" + str(k))
            out_info[-1].append("didn't␣find␣a␣target␣in␣the␣list!")
        while looping:
```

```
# calculate the shift
deltarLength = math.sqrt((groups[j][0][8]**2.0) + (
    groups[j][0][9]**2.0) + (groups[j][0][10]**2.0))
xdir = groups[j][min_pair_index][8]/deltarLength
ydir = groups[j][min_pair_index][9]/deltarLength
zdir = groups[j][min_pair_index][10]/deltarLength
shiftx = k * shift_value * xdir
shifty = k * shift_value * ydir
shiftz = k * shift_value * zdir
newx = groups[j][min_pair_index][2] + shiftx
newy = groups[j][min_pair_index][3] + shifty
newz = groups[j][min_pair_index][4] + shiftz
newdx = groups[j][min_pair_index][5] - newx
newdy = groups[j][min_pair_index][6] - newy
newdz = groups[j][min_pair_index][7] - newz
newDist = math.sqrt((newdx**2.0) + (newdy**2.0) + (newdz
    **2.0))
#check if its a valid shift, i.e. >= "r_min" and not too
     close to any other atom either
if isPositionValid(target_id, [newx,newy,newz], pairs,
    min_pair_dists):
     num_shifted_structs += 1
     # open outfile for writing
     outfile_name = str(num_shifted_structs) + ".PSI.in"
     outfile = open(outfile_name, "w")
     # add info
     # file  pair_id  r1_old  r1_shift  r1_new  r2
        old_dist  new_dist  delta_dist  min_dist
     out_info.append([])
     if k == 1:
          out_info[-1].append("* struct-" + str(i) + "-" +
              str(j) + "-" + str(k))
     else:
          out_info[-1].append("  struct-" + str(i) + "-" +
              str(j) + "-" + str(k))
     out_info[-1].append(groups[j][min_pair_index][0])
     out_info[-1].append(f"{groups[j][min_pair_index
        ][2]:.2f}" + "," + f"{groups[j][min_pair_index
        ][3]:.2f}" + "," + f"{groups[j][min_pair_index
        ][4]:.2f}")
     out_info[-1].append(f"{shiftx:.2f}" + "," + f"{
        shifty:.2f}" + "," + f"{shiftz:.2f}")
     out_info[-1].append(f"{newx:.2f}" + "," + f"{newy:.2
        f}" + "," + f"{newz:.2f}")
     out_info[-1].append(f"{groups[j][min_pair_index
        ][5]:.2f}" + "," + f"{groups[j][min_pair_index
        ][6]:.2f}" + "," + f"{groups[j][min_pair_index
        ][7]:.2f}")
     out_info[-1].append(f"{groups[j][min_pair_index
        ][11]:.2f}")
     out_info[-1].append(f"{newDist:.2f}")
```

```
                out_info[-1].append(f"{newDist␣-␣groups[j][
                    min_pair_index][11]:.2f}")
                out_info[-1].append(f"{min_pair_dists[j][1]:.2f}")
                # print the structure
                for l in range(target_qe_line_ind):
                    print(qe_out_lines[l], file=outfile)
                # insert the new structural details
                print("CELL_PARAMETERS␣{angstrom}", file=outfile)
                print(str(cell_a[0]) + "␣␣" + str(cell_a[1]) + "␣␣"
                    + str(cell_a[2]), file=outfile)
                print(str(cell_b[0]) + "␣␣" + str(cell_b[1]) + "␣␣"
                    + str(cell_b[2]), file=outfile)
                print(str(cell_c[0]) + "␣␣" + str(cell_c[1]) + "␣␣"
                    + str(cell_c[2]), file=outfile)
                print("ATOMIC_POSITIONS␣{angstrom}", file=outfile)
                for l in range(num_atoms):
                    if i == l:
                        # this atom is the one to move
                        outline = species_labels[0] + "␣␣␣␣"
                        outline += str(newx) + "␣␣␣␣"
                        outline += str(newy) + "␣␣␣␣"
                        outline += str(newz)
                        print(outline, file = outfile)
                    else:
                        # this atom stays in its original position
                        print(output_lines[l], file=outfile)
                # print the rest of the qe template
                for l in range(target_qe_line_ind + 1, len(
                    qe_out_lines)):
                    print(qe_out_lines[l], file=outfile)
                outfile.close()
                # increment k
                k += 1
            else:
                # if the position isn't valid, make a note
                out_info.append([])
                out_info[-1].append("***␣struct-" + str(i) + "-" +
                    str(j) + "-" + str(k))
                out_info[-1].append("new␣position␣is␣too␣close␣to␣
                    another␣atom!")
                # check if we should stop looping
                if newDist < min_pair_dists[j][1]:
                    # stop looping
                    looping = False
                else:
                    # if we should not stop looping yet, increment k
                    k += 1

# output info about run
outfile = open("info.out", "w")
print("a␣=␣" + str(cell_a), file=outfile)
```

```
print("b = " + str(cell_b), file=outfile)
print("c = " + str(cell_c), file=outfile)
print("num_species = " + str(num_species), file=outfile)
print("species_labels = " + str(species_labels), file=outfile)
print("species_num_atoms = " + str(species_num_atoms), file=outfile)
print("pair_types = " + str(pair_types), file=outfile)

print ("Total number of generated structures: " + str(
    num_shifted_structs), file=outfile)
print("file\tpair_id\tr1_old\tr1_shift\tr1_new\tr2\told_dist\
    tnew_dist\tdelta_dist\tmin_dist", file=outfile)
for line in out_info:
    outstr = ""
    for col in line:
        outstr+= col + "    "
    print(outstr, file=outfile)
outfile.close()

# make directory for the QE runs and move files there
os.system("mkdir scf_runs")
os.system("mv *PSI.in scf_runs")
```

Appendix B: Publication: Computational investigation of the structural and electrolyte properties of the extended family of lithium (thio)boracite materials: $Li_4B_7O_{12}Cl$ and beyond

Computational investigation of the structural and electrolyte properties of the extended family of lithium (thio)boracite materials: $Li_4B_7O_{12}Cl$ and beyond

D. Cory Lynch [1,*] Yan Li,[2,*] Pieremanuele Canepa,[2,3,4] and N. A. W. Holzwarth [5]

[1]*Department of Physics, Wake Forest University, Winston-Salem, North Carolina 27109, USA*
[2]*Department of Materials Science and Engineering, National University of Singapore, Singapore 119077, Singapore*
[3]*Department of Electrical and Computer Engineering, University of Houston, Houston, Texas 77204, USA*
[4]*Texas Center for Superconductivity at the University of Houston (TcSUH), Houston, Texas 77204, USA*
[5]*Department of Physics, Wake Forest University, Winston-Salem, North Carolina 27109, USA*

(Received 13 December 2023; revised 5 March 2024; accepted 15 May 2024; published 3 June 2024)

Motivated by previous investigations that showed very promising high ionic conductivities within relatively stable framework structures, we report a systematic first-principles study of the extended family of lithium (thio)boracites, consisting of eight chemical compositions. Three of the compositions—$Li_4B_7O_{12}Cl$, $Li_4Al_3B_4O_{12}Cl$, and $Li_6B_7S_{13}Cl$—are comparable to synthesized and analyzed materials reported in the experimental literature. The five additional compositions—$Li_4B_7S_{12}Cl$, $Li_4Al_3B_4S_{12}Cl$, $Li_6B_7O_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$—are predicted from the computational modeling and analysis presented in this paper. For each material, idealized ordered rhombohedral, cubic, or monoclinic ground-state structures are determined. Through various methodologies including thermodynamic, voltage window, and harmonic phonon analyses, stability is assessed for all eight Li (thio)boracite-derived compositions. Based on the dominant energetics of density functional theory, an analysis of the thermodynamically accessible phases predicts stability for $Li_4B_7O_{12}Cl$ only. The analysis of the voltage windows of these materials suggests that the sulfur materials are much more reactive in contact with lithium metal than their oxygen counterparts. This reactivity problem has been identified in other highly conducting solid electrolytes and various mitigation methods discussed in the literature look promising. Within the harmonic approximation, phonon analysis predicts that all eight materials are dynamically stable in their ground-state structures. Future investigations will focus on the performance of the family of lithium (thio)boracites, including ionic conductivity predictions and ion migration mechanisms.

DOI: 10.1103/PhysRevMaterials.8.065401

I. INTRODUCTION

Recently, there has been considerable interest in lithium boracite materials as fast ion conductors with possible application to all-solid-state battery technology. These materials are characterized by high-symmetry frameworks with void channels available for ion conduction. For the Li_4 boracites, a prototypical material was studied in the 1970s [1] with the composition $Li_4B_7O_{12}Cl$. More recently, Tan and coworkers [2] developed an efficient synthesis method and improved the room-temperature ionic conductivity of this material by two orders of magnitude over the conductivity reported by Jeitschko and coworkers [1]. Our group [3] computationally examined the structural properties and ion migration mechanisms of $Li_4B_7O_{12}Cl$. A related Li_4 boracite has been synthesized with aluminum substitutions on some of the boron sites by Kajihara and coworkers [4] and by Katsumata and coworkers [5] to form $Li_4Al_3B_4O_{12}Cl$, and has been found to have favorable ion conducting properties. In addition to the family of Li_4 boracite materials, evidence for an even more highly conducting "stuffed Li" thioboracite (Li_6) family of materials has appeared with the report in 2021 by Kaup and coworkers [6] of the synthesis and characterization of $Li_6B_7S_{13}I$.

Here we report the results of a detailed computational study of four combinations of components, considering both Li_4 and Li_6 stoichiometries, focusing on eight members of the Li (thio)boracite-derived family. Structural optimizations were performed, assuming ordered structures related to or derived from experimentally analyzed space groups. In addition to the three experimentally identified compounds—$Li_4B_7O_{12}Cl$, $Li_4Al_3B_4O_{12}Cl$, and $Li_6B_7S_{13}Cl$, five additional stoichiometries—$Li_6B_7O_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, $Li_4B_7S_{12}Cl$, $Li_6B_7S_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$—are predicted from simulations. The optimized structures and their phonon spectra are examined, as well as their synthesizability in terms of a convex hull construction and of the estimation of the electrochemical voltage windows.

The paper is organized as follows. Section II outlines the main computational methods used in this paper. Section III presents the idealized structural forms found by static lattice optimizations including four main optimized ordered structures as well as some additional optimized ordered structures having energies above that of the global minimum. Section IV examines the phase diagram of the Li (thio)boracite family in terms of a convex hull analysis while Sec. V extends the analysis of the static lattice formation energy results to infer the stability of these materials in battery conditions in terms of

*These authors contributed equally to this work.

 ©2024 American Physical Society

estimation of voltage windows. Section VI presents the results of harmonic phonon analysis of the optimized structures of the eight electrolyte candidates. A summary and conclusions are presented in Sec. VII.

II. CALCULATIONAL METHODS

All calculations were performed using the Wake Forest University High Performance Computing Facility [7]. The computational methods were based on density functional theory (DFT) [8,9] and density functional perturbation theory (DFPT) [10–14], using the projector augmented wave (PAW) [15] formalism. The modified generalized gradient formulation known as PBEsol [16] was used to treat the exchange and correlation effects. The PAW basis and projector functions for Li, B, O, Al, S, and Cl were generated by the ATOMPAW code [17]. Simulations for idealized models of the studied crystals, usually in their primitive unit cells, were carried out using the ABINIT [18,19] and the QUANTUM ESPRESSO packages [20,21]. For a given stoichiometry of N_i atoms of each element i, results from optimizing the lattice structure of ordered arrangements of lithium ions within the (thio)boracite-derived frameworks provide structurally optimized DFT total electronic energies. As in our previous computational study on solid electrolytes [22], thanks to the Born-Oppenheimer approximation [23] where the atomic coordinates are treated as "static" classical point particles, and to the electronically insulating nature of these materials, it is reasonable to identify the optimized DFT total electronic energy as a good approximation to the static lattice internal energy $U_{SL}(\{N_i\})$. Well-converged plane-wave expansions—$|\mathbf{k}+\mathbf{G}|^2 \leqslant 70\,\text{Bohr}^{-2}$ and $|\mathbf{k}+\mathbf{G}|^2 \leqslant 81\,\text{Bohr}^{-2}$—were used with ABINIT and QUANTUM ESPRESSO, respectively. For approximating integrals over the Brillouin zone, uniform midpoint sampling was used with sampling volumes less than 10^{-3} (radians/Bohr)^3. Structural optimizations were carried out by using the conjugate gradient based algorithms implemented in the two codes, with residual forces magnitudes limited to less than 10^{-4} Ryd/Bohr. In this way, we find total energy differences between the two codes to be less than 0.002 eV/FU [eV/(formula unit) or 0.0001 eV/atom] for the materials in this study. In Secs. III, IV, and V, we focus on the static lattice internal energies $U_{SL}(N_i)$ of the materials, approximating the Helmholtz free energy as

$$F(T, V, \{N_i\}) \approx U_{SL}(\{N_i\}), \tag{1}$$

where T and V denote temperature and volume, respectively. Similar approaches have been found to be useful for the development of large materials databases such as the Materials Project [24] and the Open Quantum Materials Database [25,26]. Generally, lattice vibrations within the harmonic approximation, considered in Sec. VI, and other temperature-dependent effects, while important, are estimated to have energetic contributions of 0.05 eV/FU or less.

The present paper also benefited from the use of several public domain analysis tools such as FINDSYM [27] for identifying the space group structure of optimization outputs, VESTA [28] for structural visualization, and XMGRACE [29] for plotting.

III. IDEALIZED STRUCTURES

Two important components of the structural forms of the lithium (thio)boracite family of materials are the (thio)borate framework and the void regions containing the mobile Li ions and Cl or other halide counter ions. The framework structure is negatively charged, carrying a formal charge of -3 ($[B_7O_{12}]^{3-}$ for example) or -5 ($[B_7O_{13}]^{5-}$ for example) for the Li_4 or Li_6 materials, respectively. It is flexible enough to accommodate interconnected, corner-sharing anions including tetrahedral BO_4, AlO_4, BS_4, or AlS_4 units and, in some cases, planar triangular BO_3 or BS_3 units. Typically the halide ions occupy well-defined positions in the voids while the Li ions occupy various other sites in the void regions.

The experimental literature on Li (thio)boracite materials identifies their structures with the face-centered cubic $F\bar{4}3c$ (No. 219) [30] space group. This includes both the Li_4 and Li_6 materials [1,2,4–6,31–35], characterized with fractional occupancy on the Li sites. The beauty of these structures is that the $F\bar{4}3c$ space group allows for natural interstitial sites available for Li ion migration, which is consistent with the fractional occupancy deduced from diffraction analysis. In our previous study [3] of $Li_4B_7O_{12}Cl$, static lattice optimization results identified the existence of a fully ordered and more stable structure of face-centered $R3c$ (No. 161). Interestingly, from the viewpoint of group theory analysis, the rhombohedral structure could be predicted by noting that the $R3c$ space group is a simple subgroup of $F\bar{4}3c$. For the extended family of Li (thio)boracites, in addition to the $R3c$ structure, two other ordered structural forms, characterized by the $F\bar{4}3c$ and Cc space groups have been found to characterize the optimized ground-state structures for one or more of the eight members.

In order to present an overview of the results, the optimized lattice and other structural parameters in their conventional cell settings [30] together with the static lattice total energy differences from the density functional calculations are listed in Table I. While a systematic study of phase stability is presented in Sec. IV, here we focus on the comparisons within each of the four combinations of components, including both Li_4 and Li_6 stoichiometries. For this purpose, for each family of compositions, it is convenient to define the static lattice energy difference based on the $R3c$ structure of the Li_4 compound chosen as the reference. For family members having the Li_4 stoichiometries, $\Delta U_{SL}(\text{Li}_4\text{boracite})$ is defined as

$$\begin{aligned}\Delta U_{SL}(\text{Li}_4\text{boracite}) \equiv\ & U_{SL}(\text{Li}_4\text{boracite}) \\ & - U_{SL}(\text{Li}_4\text{boracite}[R3c]).\end{aligned} \tag{2}$$

For family members having the Li_6 stoichiometries, $\Delta U_{SL}(\text{Li}_6\text{boracite})$ is defined as

$$\begin{aligned}\Delta U_{SL}(\text{Li}_6\text{boracite}) \equiv\ & U_{SL}(\text{Li}_6\text{boracite}) \\ & - U_{SL}(\text{Li}_4\text{boracite}[R3c]) - U_{SL}(\text{Li}_2\text{X}).\end{aligned} \tag{3}$$

Here Li_2X represents the static lattice energy of one formula unit of $Fm\bar{3}m$ Li_2O or of $Fm\bar{3}m$ Li_2S as appropriate. These energies are listed in Table I in the column labeled ΔU_{SL} in units of eV/FU.

Here we see that for the $R3c$ structures, the conventional cell rhombohedral angle deviates from 90 degrees by less

TABLE I. Summary of the structural parameters and relative energies for the dominant structural forms of the (thio)boracites considered in this study. Here the column "SG" gives the space group based on the conventional cells listed in the crystallography tables [30]. The column "Lattice" lists the conventional cell lattice parameters in Angström units and "Angles" lists the lattice angles in degrees with the measured values in parentheses when available from the literature. The conventional cells for the $F\bar{4}3c$ and $R3c$ structures contain eight formula units while the conventional cells for the Cc structures contain four formula units. For all three structures, the primitive cells all contain two formula units; the Cartesian coordinates of the primitive lattices and the corresponding fractional atomic coordinates are all given in the Part I of the SM [36]. The column Tri/Tet parameterizes the (thio)borate framework anions in terms of the numbers of interconnected "triangular planar" vs "tetrahedral", (BO_3 or BS_3) vs (BO_4, BS_4 AlO_4, or AlS_4) framework components in a primitive cell. The column "Vol" lists the volume per formula unit in Angström^3 units. The column ΔU_{SL} lists the static lattice energy difference in eV/FU for each family of materials as described by Eqs. (2) and (3).

Formula	SG	Lattice	Angle	Tri/Tet	Vol	ΔU_{SL}
$Li_4B_7O_{12}Cl$	$R3c$	$a = 12.1$ (12.1410)[a]	$\alpha = 90.1$ (90.0)[a]	4/3	223.0	0.00
$Li_4B_7O_{12}Cl$	$F\bar{4}3c$	$a = 12.2$	$\alpha = 90.0$	4/3	224.8	0.21
$Li_6B_7O_{13}Cl$	$R3c$	$a = 12.2$	$\alpha = 89.4$	1/6	224.3	–1.58
$Li_6B_7O_{13}Cl$	Cc	$a = 15.0$	$\alpha = 90.0$	1/6	225.8	–1.26
		$b = 8.6$	$\beta = 124.7$			
		$c = 8.6$	$\gamma = 90.0$			
$Li_4Al_3B_4O_{12}Cl$	$R3c$	$a = 13.0$ (12.9687)[b]	$\alpha = 91.1$ (90.0)[b]	4/3	275.6	0.00
$Li_4Al_3B_4O_{12}Cl$	$F\bar{4}3c$	$a = 13.0$	$\alpha = 90.0$	4/3	272.0	1.19
$Li_6Al_3B_4O_{13}Cl$	$R3c$	$a = 13.0$	$\alpha = 88.3$	1/6	275.0	–0.22
$Li_6Al_3B_4O_{13}Cl$	Cc	$a = 16.0$	$\alpha = 90.0$	1/6	273.8	–0.40
		$b = 8.9$	$\beta = 127.1$			
		$c = 9.7$	$\gamma = 90.0$			
$Li_4B_7S_{12}Cl$	$R3c$	$a = 14.9$	$\alpha = 89.8$	4/3	415.7	0.00
$Li_4B_7S_{12}Cl$	$F\bar{4}3c$	$a = 14.9$	$\alpha = 90.0$	4/3	414.6	–0.05
$Li_6B_7S_{13}Cl$	$R3c$	$a = 15.1$ (15.245)[c]	$\alpha = 89.2$ (90.0)[c]	0/7	431.3	–0.96
$Li_6B_7S_{13}Cl$	Cc	$a = 18.5$	$\alpha = 90.0$	0/7	429.9	–1.05
		$b = 10.5$	$\beta = 124.6$			
		$c = 10.8$	$\gamma = 90.0$			
$Li_4Al_3B_4S_{12}Cl$	$R3c$	$a = 16.1$	$\alpha = 89.6$	4/3	516.9	0.00
$Li_4Al_3B_4S_{12}Cl$	$F\bar{4}3c$	$a = 15.8$	$\alpha = 90.0$	4/3	489.0	0.85
$Li_6Al_3B_4S_{13}Cl$	$R3c$	$a = 16.1$	$\alpha = 87.8$	0/7	519.7	–0.34
$Li_6Al_3B_4S_{13}Cl$	Cc	$a = 19.8$	$\alpha = 90.0$	0/7	508.2	–0.65
		$b = 10.8$	$\beta = 126.1$			
		$c = 11.8$	$\gamma = 90.0$			

[a]Reference [1] for $Li_4B_7O_{12}Cl$ in the $F\bar{4}3c$ structure (with fractional occupancy). Similar results are reported in Ref. [5].
[b]Reference [5] for $Li_4Al_3B_4O_{12}Cl$ in the $F\bar{4}3c$ structure (with fractional occupancy).
[c]Reference [6] for $Li_6B_7S_{13}I$ in the $F\bar{4}3c$ structure (with fractional occupancy).

than 3 degrees, indicating that the rhombohedral distortions relative to cubic are small. For the three materials that can be compared with the experimental literature, the corresponding fractional coordinates are listed in Appendix A while the full structural parameters of the primitive cells are given in the Part I of the Supplemental Material (SM) [36]. In order to widen the scope of structural features, Table I also parameterizes the (thio)borate framework in terms of the numbers of interconnected "triangular planar" (BO_3 or BS_3) versus "tetrahedral" (BO_4, BS_4, AlO_4, or AlS_4) framework components in a primitive cell in the column labeled "Tri/Tet" and also the volume per formula unit in units of Å^3 in the column labeled "Vol".

Details for the $R3c$, $F\bar{4}3c$, and Cc structural forms are presented in Secs. III A, III B, and III C, respectively. A summary of additional structural analyses is presented in Sec. III D.

A. Structures characterized by the *R3c* space group

There are many equivalent rhombohedral lattice conventions in the literature. In this paper, we follow the convention described in Ref. [3] since it most easily reveals the structural relationship with the related face-centered-cubic structure. In performing the static lattice structural analysis of the eight members of our Li (thio)boracite family, we find optimized ordered $R3c$ structures for all of them. For the Li_4 (thio)boracites, the rhombohedral distortion (Ref. [3]) follows from the low-temperature single crystal analysis of $Li_4B_7O_{12}Cl$ results of Jeitschko *et al.* [1]. The conventional cell of this crystal has 32 Li sites. At low temperatures, 24 Li sites with Wyckoff label 24*c* are fully occupied while 25% of the sites with Wyckoff label 32*e* are occupied. This partial occupancy causes the rhombohedral distortion to the $R3c$ space group, which is a subgroup of the $F\bar{4}3c$ space group. In this case, the conventional face-centered $R3c$ structure fully accommodates the eight Li sites with Wyckoff label 8*a*. The remaining 24 empty Li sites are designated by the Wyckoff label 24*b*, which are available as "natural interstitial" sites, participating in the Li ion mobility.

Table I shows that the structural parameters computed for $R3c$ $Li_4B_7O_{12}Cl$ are very close to experimental results

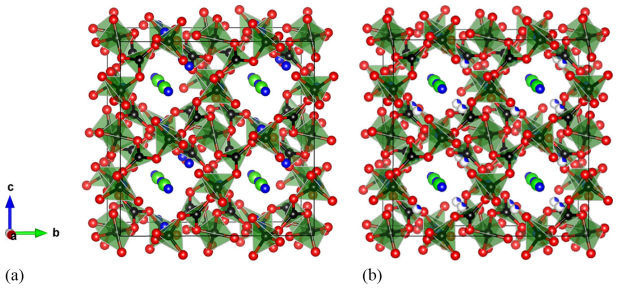


FIG. 1. Visualization of a conventional unit cell of $Li_4B_7O_{12}Cl$ in (a) its computed optimized structure and (b) the experimental structure of Ref. [1]. Li, B, O, and Cl are represented with blue, black, red, and light green balls; green shaded polyhedra indicate the BO_3 and BO_4 framework components. The fractionally occupied Li sites in the experimental structure are indicated with balls shaded 25% blue and 75% white.

deduced from the diffraction analysis of the $F\bar{4}3c$ structure [1]. The slight rhombohedral distortion, with the conventional cell rhombohedral angle of 90.1 degrees compared with ideal cubic angle of 90 degrees, is also evident in Fig. 1; the conventional cell fractional coordinates from the optimization are compared with the corresponding results from experiment in Appendix A, while the full structural parameters of the primitive cell for the optimized structure are given in Part I of the SM [36].

Similarly, Table I shows that the structural parameters computed for $R3c$ $Li_4Al_3B_4O_{12}Cl$ are very close to experimental results deduced from the diffraction analysis of the $F\bar{4}3c$ structure [4]. These results are visualized in Fig. 2; the conventional cell fractional coordinates from the optimization are compared with the corresponding results from experiment in Appendix A, while the full structural parameters of the primitive cell for the optimized structure are given in the Part I of the SM [36]. In this case, we see that the computed rhombohedral distortion is again small, with the conventional

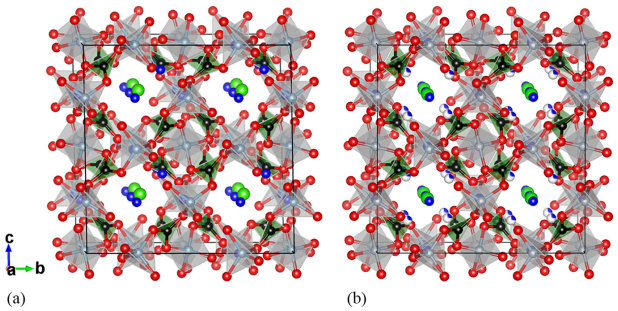


FIG. 2. Visualization of a conventional unit cell of $Li_4Al_3B_4O_{12}Cl$ in (a) its computed optimized structure and (b) the experimental structure of Ref. [4]. Li, Al, B, O, and Cl are represented with blue, grey, black, red, and light green balls; grey shaded tetrahedra indicate AlO_4 components and dark green shaded triangles indicate the BO_3 components of the framework. The fractionally occupied Li sites in the experimental structure are indicated with balls shaded 25% blue and 75% white.

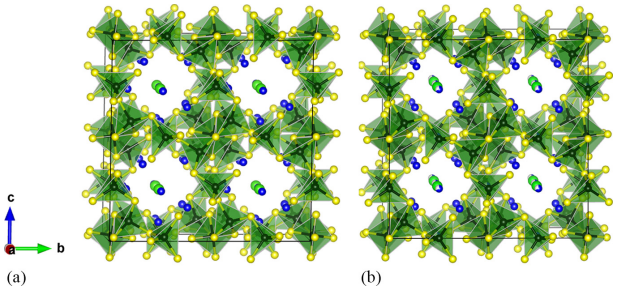


FIG. 3. Visualization of a conventional unit cell of $Li_6B_7S_{13}Cl$ in (a) its computed optimized $R3c$ structure and (b) the experimental structure of $Li_6B_7S_{13}I$ analyzed in Ref. [6]. Li, B, S, and Cl (or I) are represented with blue, grey, black, yellow, and light green balls; dark green shaded tetrahedra indicate BS_4 components of the framework. The fractionally occupied Li sites in the experimental structure are indicated with balls shaded 75% blue and 25% white.

cell rhombohedral angle of 91.1 degrees, compared with the ideal cubic angle of 90 degrees.

A similar group theory analysis can be applied to the Li_6 (thio)boracites. In this case, the $F\bar{4}3c$ conventional cell, if we assume that Li sites with Wyckoff label 24c are fully occupied, the remaining Li's would partially occupy the sites with Wyckoff labels 32e at the level of 75%. With the rhombohedral distortion, the conventional cell of the $R3c$ structure accommodates the Li sites with Wyckoff label 24b while the remaining eight empty Li sites are associated with the Wyckoff label 8a, which are available as "natural interstitial" sites, which facilitate Li ion mobility. These results are visualized in Fig. 3; the conventional cell fractional coordinates from the optimization are compared with the corresponding results from experiment in Appendix A, while the full structural parameters of the primitive cell for the optimized structure are given in the Part I of the SM [36]. In this case, the rhombohedral distortion is small, with rhombohedral angle of 89.2 degrees compared with 90 degrees for ideal cubic structure. The structure is visualized in Fig. 3 and the fractional coordinates are given in Table VI. As pointed out in Sec. VI, harmonic phonon analysis finds that this structure has some imaginary modes in a small volume of the Brillouin zone near $\mathbf{q} = 0$, indicating some dynamic instability. Further, as pointed out in Sec. III C, it is possible to find a lower-energy ordered structural form for $Li_6B_7S_{13}Cl$ than the $R3c$ structure. Interestingly, the experimental diffraction analysis of this structure by Kaup and coworkers [6] visualized in Fig. 3 finds a different fractional occupancy pattern for the $F\bar{4}3c$ cubic structure. They find that Li ions fully occupy the sites with Wyckoff labels 32e while partially occupying the sites with Wyckoff labels 24c at the level of 66.7%.

An interesting wrinkle to the structural analysis of $Li_6B_7S_{13}Cl$, is the possibility of a tetragonal phase, which Kaup and coworkers [6] identified in their experimental and modeling work. Our static lattice density functional analysis also found such a structure with $\Delta U_{SL} = -1.00$ eV/FU as defined in according to Eqs. (2) and (3). However, in the lattice vibrational analysis of this structure, we found imaginary phonon modes throughout the Brillouin zone for this structure, as did Kaup and coworkers [6]. Because of the dynamic

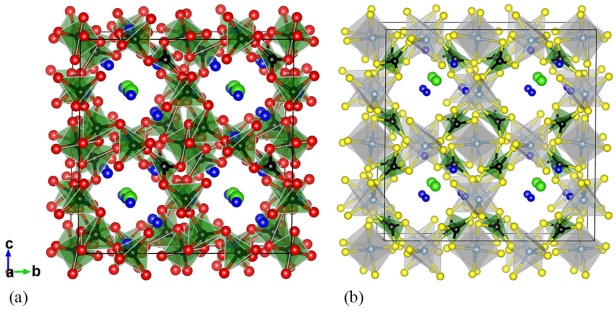


FIG. 4. Visualization of conventional unit cells of (a) $Li_6B_7O_{13}Cl$ and (b) $Li_4Al_3B_4S_{12}Cl$ in their computed optimized $R3c$ structures. Li, Al, B, O, S, and Cl are represented with blue, grey, black, red, yellow, and light green balls; dark green shaded polyhedra indicate BO_3/BS_3 and BO_4 components of the framework and the AlS_4 components are indicated with grey tetrahedra.

instability of this structure, it was not considered further in this paper.

Of the predicted but not yet experimentally realized members of the lithium boracite family, $Li_6B_7O_{13}Cl$ and $Li_4Al_3B_4S_{12}Cl$ are found to optimize in the $R3c$ structure. Their conventional unit cells are visualized in Fig. 4.

B. Structures characterized by the $F\bar{4}3c$ space group

The $F\bar{4}3c$ structure is another structural motif that logically follows from experimental evidence. The group theory analysis suggests that the Li_4 (thio)boracites can be ordered if the 32 Li ions fully occupy the sites with Wyckoff label 32e. As shown in Table I, this structure is energetically favored for $Li_4B_7S_{12}Cl$, but not for the other Li_4 family members. The $R3c$ and $F\bar{4}3c$ structures of $Li_4B_7S_{12}Cl$ are visualized in Fig. 5.

C. Structures characterized by the Cc space group

Another structural motif follows from the study of Kaup and coworkers [6] who also observed $Li_6B_7S_{13}I$ in a lower-symmetry structural form. Rather than the tetragonal structure

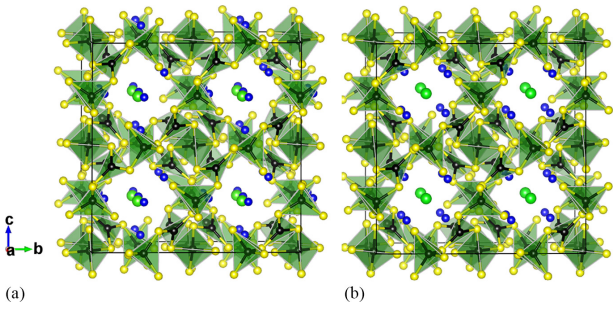


FIG. 5. Visualization of a conventional unit cell of $Li_4B_7S_{12}Cl$ in (a) its computed optimized structure with $R3c$ symmetry and (b) its computed optimized structure with $F\bar{4}3c$ symmetry, found to be energetically favorable. Li, B, S, and Cl are represented with blue, grey, black, yellow, and light green balls; dark green shaded tetrahedra indicate BS_4 components of the framework.

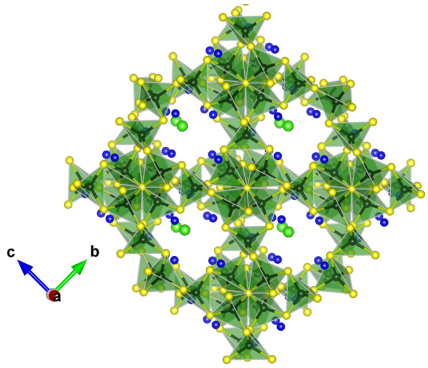


FIG. 6. Structure of $Li_6B_7S_{13}Cl$ in its computed optimized structure with Cc symmetry, visualized using multiples of the nearly orthorhombic cell defined in Eq. (5). Li, B, S, and Cl are represented with blue, grey, black, yellow, and light green balls; dark green shaded tetrahedra indicate BS_4 components of the framework.

suggested by simulations and experimental analysis of their study, we found a lower-energy centered monoclinic structure with the space group Cc (No. 9) [30]. The discovery of the Cc structure was initially made by performing structural optimization of configurations of $Li_6B_7S_{13}Cl$ generated during molecular dynamics simulations. The conventional cell of this base-centered monoclinic structure, consistent with Table I is given in cartesian coordinates by

$$\mathbf{a}_c = a\hat{\mathbf{x}} \quad \mathbf{b}_c = b\hat{\mathbf{y}} \quad \mathbf{c}_c = c(\cos\beta\hat{\mathbf{x}} + \sin\beta\hat{\mathbf{z}}). \tag{4}$$

For illustrating the similarities to the framework and void structures found in other (thio)boracites, a nearly orthorhombic supercell having the same volume and number of atoms (108) can be constructed with lattice parameters

$$\mathbf{a}_s = \mathbf{a}_c + \mathbf{c}_c, \quad \mathbf{b}_s = \mathbf{b}_c, \quad \mathbf{c}_s = \mathbf{c}_c. \tag{5}$$

This is shown in Fig. 6.

As indicated in Table I, we have found Cc structures for all of Li_6 (thio)boracites of this study. Interestingly, except for $Li_6B_7O_{13}Cl$, these Cc structures are computed to be their most stable structural forms. It is also interesting to note that the optimized structures for $Li_6B_7O_{13}Cl$ have a smaller volume per formula unit in the $R3c$ phase relative to the Cc phase, while $Li_6B_7S_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$ all have a smaller volume per formula unit in the Cc phase relative to the $R3c$ phase. To our knowledge, these Cc structures have not yet been observed for any of these materials. The predicted Cc structures of $Li_6Al_3B_4O_{13}Cl$ and $Li_6Al_3B_4S_{13}Cl$ are visualized in Fig. 7.

D. Search for additional structures

With a view toward extending the structural analysis of Li (thio)boracite family members, we conducted a few additional structural searches, focusing on the possibility of forming ordered stable structures with monoclinic (Cc) symmetry for the Li_4 (thio)boracites and on the possibility of forming order stable structures with cubic ($F\bar{4}3c$) symmetry for the Li_6 (thio)boracites.

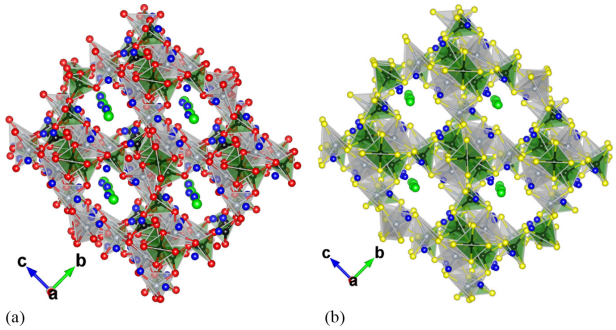


FIG. 7. Structures of $Li_6Al_3B_4O_{13}Cl$ (a) and $Li_6Al_3B_4S_{13}Cl$ (b) as computed optimized structure with Cc symmetry, visualized using multiples of the nearly orthorhombic cell defined in Eq. (5). Li, B, O, S, and Cl are represented with blue, grey, black, red, yellow, and light green balls; shaded planar triangles and tetrahedra indicate BO_3, BO_4, BS_4 (dark green), or AlS_4 (grey) components of the framework.

In the search for the Li_4 (thio)boracites with Cc symmetry, we examined each of the optimized Li_6 (thio)boracites in their Cc structure and systematically considered the distinct ways to remove one unit of Li_2O or Li_2S per formula unit while maintaining the Cc symmetry. It is convenient to work with primitive cells, which contain 2 formula units of the Li (thio)boracite and for which the Cc symmetry requires each distinct site to be paired with another symmetry related site. The distinct choices then involve finding 1 pair of symmetry related O or S atoms and two pairs of symmetry related Li atoms. By comparing the analysis of the experimental x-ray patterns based on the $F\bar{4}3c$ structure with fractional occupancy of the Li sites for $Li_4B_7O_{12}Cl$ [1] and for $Li_6B_7S_{13}Cl$ [6], the choice of the symmetry related O/S atoms to be removed could be reasonably associated with Wyckoff label a, missing in the Li_4 structure but occupied in the Li_6 structure. In addition to this fixed choice, the removed pair of O/S atoms (as adjusted to their equivalent locations in the Cc structure), two pairs of Li sites from among six distinct pairs had to be chosen for removal to make 15 distinct candidate configurations. For each of the Li_4 (thio)boracite compositions and each of their 15 candidate configurations, density functional optimizations were performed within the Cc symmetry constraint. The results find that lowest energy Li_4 (thio)boracite materials having the Cc structure have higher energy than their $R3c$ counterparts. For $Li_4B_7S_{12}Cl$, $U_{SL}^{Cc} - U_{SL}^{R3c} = +0.05$ eV/FU. Interestingly, as mentioned above, the ground-state structure of this composition has an ordered $F\bar{4}3c$ structure with $U_{SL}^{F\bar{4}3c} - U_{SL}^{R3c} = -0.05$ eV/FU. For the other three compositions, $U_{SL}^{Cc} - U_{SL}^{R3c}$ is considerably larger, ranging from 0.32 to 1.91 eV/FU.

In the search for the Li_6 (thio)boracites with ordered $F\bar{4}3c$ structures, we also performed density functional optimizations on a number of initial configurations, within the $F\bar{4}3c$ symmetry. Here, the (thio)boracite framework was initialized with a knowledge of corresponding experimental analysis of $Li_6B_7S_{13}I$ [6], while the ordered Li candidate configurations were chosen on sites with Wyckoff labels f or g, as is consistent with the $F\bar{4}3c$ space group. Since these sites only depend on one variable, we simply performed a coarse search of $x = \{0.1, 0.2, 0.3, 0.4\}$ and relaxed the structures to find the lowest energy, again ensuring it maintained the starting $F\bar{4}3c$ symmetry. The lowest-energy $F\bar{4}3c$ Li_6 structures found from this limited search were all structures with Li fully occupying the g sites and they all had much higher static lattice energies than their corresponding $R3c$ structures, ranging from 1.5 to 4.75 eV/FU higher energy. This ensures that these structures, even if metastable, will be more likely to be found in the lower energy $R3c$ or Cc structures.

These limited structural optimization studies provide further evidence that Table I includes the lowest-energy structures, including the ground-state static lattice structures, of Li (thio)boracite family of materials of this study.

More information about the electronic structures of the 16 Li (thio)boracite materials listed in Table I is given in Fig. 1, Sec. II of Part I of the SM [36] in the form of the valence band electronic partial density of states (PDOS). (Note that in this paragraph and in the corresponding section of Part I of the SM, "PDOS" refers to the electronic partial density of states.) From these plots, it is possible to make the following qualitative observations. It is clear that all of these materials are electronically insulating as expected. Interestingly, the PDOS plots are remarkably similar, particularly within each of the four compositions considered, for their four different optimized lattice structures. For all of the structures, the PDOS contributions from occupied states of Cl are highly peaked, consistent with their nonbonding attributes due to their location in the void regions of the structures. By contrast, PDOS contributions from occupied states involving the framework components B, Al, O, and S have much broader curves, consistent with their covalent bonding attributes. Apart from contributions from the localized Cl states, the top of the valence bands of these materials are dominated by O or S contributions, consistent with filled bound states associated with the atomic $2p$ or $3p$ levels. The PDOS contributions for Li in the occupied states have negligible magnitude, consistent their description as positive Li^+ ions in these materials.

IV. EQUILIBRIUM PHASE DIAGRAMS

In this section, we consider a more comprehensive analysis of the constituents of the Li (thio)boracite family of materials in order to reveal a holistic understanding of their phase equilibria. To accomplish this goal, we employ phase diagrams as a powerful analytical tool, as previously outlined by Ong *et al.* (2008) [37]. Ideally, the evaluation of phase stability for the solid materials of interest hinges upon the Helmholtz free energy $F(T, V, \{N_i\})$ as a function of temperature T, volume V, and numbers $\{N_i\}$ of constituents i.

As shown in Sec. VI, the energetic effect of lattice vibrations in the temperature range of $0\ \text{K} \leqslant T \leqslant 300\ \text{K}$ is estimated to be less than 0.05 eV/FU for the materials under investigation. Additionally, some of the materials of interest occur in gaseous form. The internal energy of an ideal gas approximates 0.03 eV/atom. For the few metallic materials in this study, the contributions to the temperature-dependent electronic heat capacity at 300 K is also negligibly small. These three contributions collectively constitute a minor fraction of the electronic binding energies, justifying the

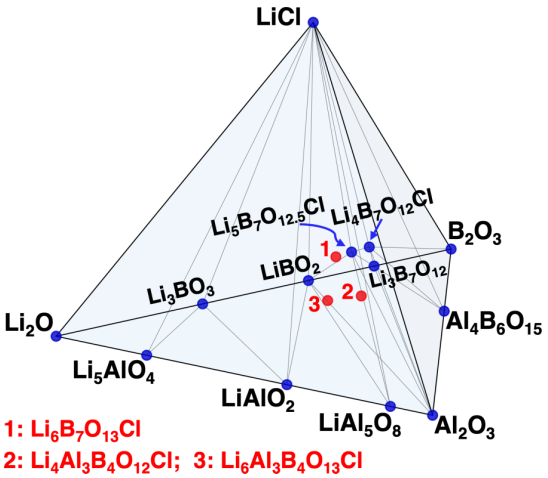


FIG. 8. Phase diagram of the quaternary subsystem of the Li-Al-B-O-Cl system, with stable phases represented by blue markers. While $Li_4B_7O_{12}Cl$ is calculated to be stable, the other three boracite materials, as listed at the bottom with numeric labels, are marked as unstable with red nodes on the diagram.

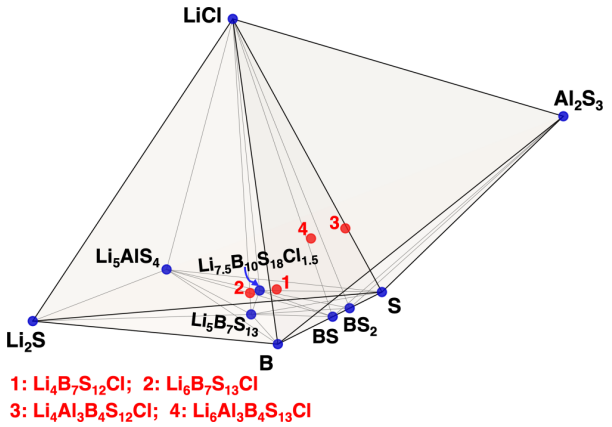


FIG. 9. Phase diagram of the pentagonal subsystem of the Li-Al-B-S-Cl system, with stable phases represented by blue markers. The four thioboracite materials, as listed at the bottom with numeric labels, are marked as unstable with red nodes on the diagram.

approximation of $F(T, V, \{N_i\})$ by its dominant electronic structure contribution given in Eq. (1). In this context, the focus is on $U_{SL}(\{N_i\})$, the static lattice energy determined from optimizing the static lattice density functional theory energy as described in Sec. II, using the QUANTUM ESPRESSO package [21] with the PBEsol [16] exchange-correlation functional.

In addition to the eight members of the Li (thio)boracite family, the elemental materials and compounds considered in this analysis were those within the composition spaces of Li-Al-B-O-Cl or Li-Al-B-S-Cl as listed in a comprehensive database—Inorganic Crystal Structure Database (ICSD) [38]. For each elemental and compound material, all the relevant initial structures were optimized to find the lowest value of $U_{SL}(\{N_i\})$. In some cases, such as for elemental O and Cl, the lowest value of $U_{SL}(\{N_i\})$ was found to correspond to a crystalline molecular crystal of dimer O_2 and Cl_2. Similar approaches have been used in the Materials Project [24] and the Open Quantum Materials Database [25,26]. The Python library Pymatgen (Python Materials Genomics) [39] was employed to construct the phase diagrams and we represent them in Figs. 8 and 9 for the boracite and thioboracite materials, respectively.

In order to quantify the energy calculations, it is convenient to introduce the concept of normalized formation energy based on the optimized static lattice energies $U_{SL}(\{N_i\})$. For example, considering the compound $LiAlB_{14}$ consisting of 16 atoms, which undergoes decomposition into its constituent elements in their standard forms,

$$LiAlB_{14} \rightarrow Li + Al + 14B, \tag{6}$$

the normalized formation energy is defined as

$$E_f(LiAlB_{14}) \equiv \frac{U_{SL}(LiAlB_{14}) - U_{SL}(Li) - U_{SL}(Al) - 14U_{SL}(B)}{16}. \tag{7}$$

All of the normalized formation energy results examined within this study are listed in Table VIII, which is available in Appendix B. This includes the complete list of materials, along with the space group of their lowest-energy structures, and the corresponding calculated normalized formation energies.

In the field of computational material science, the term "energy above the hull" $E_{\text{hull}}^{\text{above}}(\text{R})$, is commonly used to quantify the stability of compounds, distinguishing between stable and unstable phases depicted in phase diagrams [40]. This metric arises from the process of minimizing formation energies as computed by Eq. (7) within the composition space. It can be conceptualized as the energetic deviation from $E_f(\text{R})$ of compound R to the convex hull line. For a compound R that is stable and resides on the convex hull, the energy above the hull is quantified as zero,

$$E_{\text{hull}}^{\text{above}}(\text{R}) = 0. \tag{8}$$

In contrast, for an unstable compound R, the value of $E_f(\text{R})$ is greater than zero and measures the minimum reduction in formation energy for R to reach stability on the convex hull. If a stable counterpart R′ sharing the same stoichiometry as the unstable R resides on the hull, then $E_{\text{hull}}^{\text{above}}(\text{R})$ reflects the energy difference in their formation energies,

$$E_{\text{hull}}^{\text{above}}(\text{R}) = E_f(\text{R}) - E_f(\text{R}'). \tag{9}$$

In another scenario where the compound R is unstable and decomposes into a combination of stable products P_i, the energy above the hull is determined by the decomposition energy and approximated as

$$E_{\text{hull}}^{\text{above}}(\text{R}) = E_f(\text{R}) - \frac{1}{N_{\text{R}}} \sum_i x_i N_{\text{P}_i} E_f(\text{P}_i). \tag{10}$$

In this context, N_{R} and N_{P_i} denote the total number of atoms in R and the ith product P_i, respectively. The value of x_i specifies the stoichiometric coefficients for P_i in R such that $\sum_i x_i N_{\text{P}_i} = N_{\text{R}}$.

The tabulated values of $E_{\text{hull}}^{\text{above}}(\text{R})$ for compounds in the Li-Al-B-O-Cl or Li-Al-B-S-Cl composition spaces are

presented in Table VIII. The results for the eight compositions indicate that $Li_4B_7O_{12}Cl$ is the only material that is thermodynamically stable, with a hull energy of zero. Among the remaining compounds, both experimentally observed materials, $Li_4Al_3B_4O_{12}Cl$ [4,5] and $Li_6B_7S_{13}Cl$ [6], along with our predicted materials, namely $Li_6B_7O_{13}Cl$ and $Li_4B_7S_{12}Cl$, as well as the Al-doped boracite variants $Li_6Al_3B_4O_{13}Cl$, $Li_4Al_3B_4S_{12}Cl$, and $Li_6Al_3B_4S_{13}Cl$, were identified as unstable. Interestingly, for the unstable lithium boracites, both with and without aluminum substitution (entries #69–72 in Table VIII), a common decomposition product is $Li_5B_7O_{12.5}Cl$ (entry #66), which was experimentally identified in Refs. [34,41,42] and modeled in Ref. [3]. These unstable (thio)boracite materials exhibit values of $E_{\rm hull}^{\rm above}$(R) that range from 0.003 to 0.09 eV/atom, which are relatively modest. Such low values suggest that these materials could potentially achieve entropy-driven stabilization at high temperatures, analogous to the stabilization mechanisms observed in many metastable Li/Na ion conductors, where the metastability is predominantly attributed to the high intrinsic mobility of Li/Na ions within their structures. Additionally, kinetic factors, such as substantial energy barriers to phase transformations and decomposition reactions, can significantly influence the predominance of particular crystal structures. In some cases, materials with nonzero $E_{\rm hull}^{\rm above}$(R) values can be successfully synthesized by modulating the processing conditions to slow down phase transition rates, thus allowing synthesis to precede any potential decomposition. An exemplary case is provided by the experimentally synthesized $Li_4Al_3B_4O_{12}Cl$ [4], which exhibits an $E_{\rm hull}^{\rm above}$(R) value of 0.053 eV/atom.

The compound phase diagrams for Li_2O-Al_2O_3-B_2O_3-LiCl and Li_2S-Al_2S_3-B-S-LiCl systems are presented in Figs. 8 and 9 respectively, using data from Table VIII. These diagrams also offer insights into the starting materials necessary for the synthesis of specific target materials. For example, the synthesis of $Li_4B_7O_{12}Cl$ and $Li_6B_7O_{13}Cl$ would likely involve the precursors Li_2O, B_2O_3, and LiCl. Al-doped boracites can be obtained from the four binary materials located at the vertices of the phase diagram. Notably, the primary impurity phase, $LiAl_5O_8$—observed during the formation of $Li_4Al_3B_4O_{12}Cl$ by Kajihara *et al.* [4]—also appears in the phase diagram depicted in Fig. 8. For the case of thioboracites, $Li_6B_7S_{13}I$ [6] was experimentally identified using stoichiometric adjustments of Li_2S, B, S, and LiI [6]. Our calculations imply that a similar strategy is likely to work for obtaining $Li_4B_7S_{12}Cl$ and $Li_6B_7S_{13}Cl$, by using Li_2S, B, S, and LiCl as precursors. For forming Al-containing boracites, the incorporation of Al_2S_3 is suggested.

V. ESTIMATION OF VOLTAGE WINDOWS

Beyond equilibrium phase stability analysis, it is important to also consider the behavior of these materials in conditions found during battery operation. Several authors [37,39,43–48] have extended the ideas contributing to convex hull analyses to consider the effects of applied voltages and variable Li concentrations. In this case, the thermodynamics is characterized by the grand canonical potential, $\Omega(T, V, \{\mu_i\})$, which relates to the Helmholtz free energy through a Legendre transformation

$$\Omega(T, V, \{\mu_i\}) = F(T, V, \{N_i\}) - \sum_i N_i \mu_i. \tag{11}$$

Here N_i denotes the number of constituents of type i with μ_i denoting their corresponding chemical potential. These authors [37,39,43–45,48] and others developed a methodology to explore the convex hull for the grand canonical potential appropriate to these solid-state Li electrolytes. For the Li solid-state battery system, the focus is on variations in Li composition due to the thermodynamic driving forces acting on the solid electrolyte when in contact with Li reservoirs.

We can imagine a Li solid electrolyte as a reactant (R) with the possible addition of $n_{\rm Li}^a$ Li atoms from the anode reservoir such that phase equilibria (P) are formed. In this case, the reaction would be

$$\mathrm{R} + n_{\rm Li}^a \mathrm{Li} \rightarrow \mathrm{P}. \tag{12}$$

On the other hand, we can imagine a Li solid electrolyte (R) possibly forming phase equilibria (P) while depositing $n_{\rm Li}^c$ Li atoms to the cathode reservoir,

$$\mathrm{R} \rightarrow \mathrm{P} + n_{\rm Li}^c \mathrm{Li}. \tag{13}$$

In these two cases, there is a change in Li concentration $\Delta n_{\rm Li}$, which can be defined

$$\Delta n_{\rm Li} = \begin{cases} -n_{\rm Li}^a & \text{for case Eq. (12)} \\ +n_{\rm Li}^c & \text{for case Eq. (13)} \end{cases}. \tag{14}$$

In terms of the variable $\Delta n_{\rm Li}$, which signifies the possible addition of $n_{\rm Li}^a$ Li atoms from the anode reservoir or $n_{\rm Li}^c$ Li atoms from the cathode reservoir, we can approximate the grand canonical potential as

$$\Delta\Omega(\text{system}) \equiv F(\text{RHS}) - F(\text{LHS}) - \Delta n_{\rm Li}\mu_{\rm Li}. \tag{15}$$

Here RHS and LHS refer to reaction Eq. (12) or Eq. (13) as appropriate. For cases in which $\Delta n_{\rm Li} = 0$, R remains stable at $\mu_{\rm Li}$ such that P ≡ R, while for other cases, P may represent multiple compounds such as $\sum_i x_i \mathrm{P}_i$. As in Sec. IV, the Helmholtz free energies of the left (LHS) and right (RHS) sides of the reaction equations are assumed to be well represented by the static lattice internal energies $U_{SL}(\{N_i\})$ as in Eq. (1) on the optimized structure of each material and can be determined in terms of the formation energies defined in Eq. (7).

Following the work of Refs. [37,39,43–48], it is convenient to define a voltage-dependent (ϕ) Li chemical potential, $\mu_{\rm Li} \rightarrow \mu_{\rm Li}(\phi)$ according to

$$\mu_{\rm Li}(\phi) = \mu_{\rm Li}^0 - e\phi. \tag{16}$$

Here $\mu_{\rm Li}^0$ denotes the chemical potential of ideal Li metal representing a pure Li metal anode and e denotes the elementary charge. This expression is consistent with the analysis of Aydinol and coworkers [49] who analyzed the open circuit voltage $\phi^{\rm OCV}$ for a Li battery having Li chemical potentials $\mu_{\rm Li}^{\rm Cathode}$ at the cathode and $\mu_{\rm Li}^0$ at the anode,

$$e\phi^{\rm OCV} \equiv -\mu_{\rm Li}^{\rm Cathode} + \mu_{\rm Li}^0. \tag{17}$$

For a particular battery system under open circuit conditions, $\mu_{\rm Li}(\phi^{\rm OCV}) = \mu_{\rm Li}^{\rm Cathode}$ in Eq. (16). More generally, $\mu_{\rm Li}(\phi)$ in

TABLE II. For each of the eight most stable (thio)boracite electrolyte, the cathodic voltage ϕ_{CL} in units of volts (V) and the corresponding equilibrium phase compounds "CL phases", the anodic voltage ϕ_{AL} in units of volts (V) and the corresponding equilibrium phase compounds "AL phases" are listed. The last column lists the stability window ($\phi_{AL} - \phi_{CL}$) in units of volts (V).

Material	ϕ_{CL}	CL phases	ϕ_{AL}	AL phases	Window
$Li_4B_7O_{12}Cl$	1.491	$LiBO_2$, B_6O, LiCl	3.860	$Li_3B_7O_{12}$, Cl_2	2.369
$Li_6B_7O_{13}Cl$	1.357	$LiBO_2$, B_6O, LiCl	3.607	$Li_4B_7O_{12}Cl$, O_2	2.250
$Li_4Al_3B_4O_{12}Cl$	1.358	$LiBO_2$, B_6O, $LiAl_5O_8$, LiCl	3.629	$Li_4B_7O_{12}Cl$, LiCl, O_2, Al_2O_3	2.271
$Li_6Al_3B_4O_{13}Cl$	1.284	$LiBO_2$, B_6O, $LiAlO_2$	3.360	$LiBO_2$, $LiAl_5O_8$, LiCl, O_2	2.076
$Li_4B_7S_{12}Cl$	2.360	$Li_5B_7S_{13}$, BS, LiCl	2.491	BS_2, BS, LiCl	0.131
$Li_6B_7S_{13}Cl$	1.595	Li_2S, LiCl, B	2.454	BS_2, BS, LiCl	0.859
$Li_4Al_3B_4S_{12}Cl$	1.706	Li_5AlS_4, BS, Al_2S_3, LiCl	2.122	$Li_5B_7S_{13}$, Al_2S_3, LiCl, S	0.416
$Li_6Al_3B_4S_{13}Cl$	1.706	Li_5AlS_4, BS, Al_2S_3, LiCl	2.117	$Li_5B_7S_{13}$, Al_2S_3, LiCl, S	0.411

Eq. (16) allows for the exploration of possible cathode and voltage combinations for battery design and is what is used in Eq. (15). Moreover, since ideal Li metal is used as one of the reference elements in calculating the formation energies E_f according to Eq. (7), it is consistent to set $\mu^0_{\rm Li} = E_f({\rm Li}) = 0$. Consistent with this energy convention, the voltage and Li chemical potential are related by

$$e\phi = -\mu_{\rm Li}(\phi). \tag{18}$$

In these terms, we approximate the change in the grand canonical potential as

$$\Delta\Omega({\rm system}, \phi, \Delta n_{\rm Li}) = \Delta E_{gcp}({\rm system}) + \Delta n_{\rm Li} e\phi. \tag{19}$$

In this expression,

$$\Delta E_{gcp}({\rm system}) = 0 \quad \text{when} \quad \Delta n_{\rm Li} = 0, \tag{20}$$

and

$$\Delta E_{gcp}({\rm system}) \equiv \sum_i x_i N_{{\rm P}_i} E_f({\rm P}_i) - N_{\rm R} E_f({\rm R}) \quad \text{when} \quad \Delta n_{\rm Li} \neq 0. \tag{21}$$

In reporting values of $\Omega({\rm system}, \phi, \Delta n_{\rm Li})$, such as in Table IX, it is convenient to normalize the energies by the number $N_{\rm R}^{\rm NonLi}$ of non-Li atoms in the "reactant" electrolyte R. Accordingly, the normalized change in the grand canonical potential can be defined,

$$\overline{\Delta\Omega} \equiv \frac{\Delta\Omega}{N_{\rm R}^{\rm NonLi}}. \tag{22}$$

For a given solid electrolyte, exploring the change in the grand canonical potential $\Delta\Omega$ for a wide range of possible reaction products as a function of ϕ and $\Delta n_{\rm Li}$ indicates stability and reactivity information about the system based on its bulk thermodynamic energies. By this measure, the window of stability for the solid electrolyte is defined as the range of values of the potential ϕ [and the corresponding ranges of chemical potentials $-\mu_{\rm Li}(\phi)$] for $\Delta\Omega({\rm system}, \phi, \Delta n_{\rm Li}) = 0$. Conditions that produce $\Delta\Omega < 0$ indicate that a spontaneous decomposition or the exchange Li with the anode or cathode reservoir is predicted to occur. A number of voltage window estimates for electrolytes have appeared in the literature [43–45,50]. Typically the range is found to be $0 \leqslant \phi \leqslant 5$ V. The lower value ϕ_{CL} ("cathodic limit" in the chemistry literature) represents the lowest voltage at which Li from that anode ceases to react with the electrolyte ($n^a_{\rm Li} \to 0$). The upper value of ϕ_{AL} ("anodic limit" in the chemistry literature) of the range of $\Delta\Omega({\rm system}, \phi, \Delta n_{\rm Li}) = 0$ represents the highest voltage before which Li from the cathode starts to react with the electrolyte ($0 \to n^c_{\rm Li}$). Using the Pymatgen software package [39], we examined the grand canonical potential phase diagrams of the eight most stable (thio)boracite electrolytes over a voltage range of $0\ {\rm V} \leqslant \phi \leqslant 5\ {\rm V}$, with values distributed uniformly. A graphical representation of the results for the voltage windows are presented in Fig. 10, while the corresponding numerical data can be found in Table II. Further results are detailed in Appendix C. Table IX presents the complete reaction equations, accompanied by their respective normalized decomposition reaction energies $\overline{\Delta\Omega}(\phi)$ at transition voltages/Li chemical potentials. Meanwhile, Table X lists the $\overline{\Delta\Omega}(\phi)$ energies at the extreme voltages of $\phi = 0$ V and $\phi = 5$ V.

As illustrated in Fig. 10, the voltage windows of thioboracite materials are found to be relatively narrow and fall within the range of their thioboracite analogues. This observation is consistent with previous studies, indicating suboptimal voltage windows in sulfide-based electrolytes compared to oxide-based alternatives [51,52]. Specifically, boracite materials define a stability window from 2.076 V to 3.369 V. In contrast, thioboracite derivatives display a range 0.131–0.859 V. Among the studied materials, $Li_4B_7S_{12}Cl$

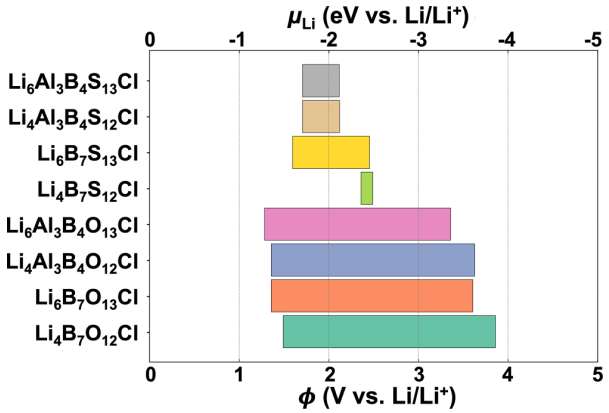


FIG. 10. The electrochemical stability ranges of (thio)boracites electrolyte materials, generated using the data summarized in Table II.

demonstrates the most constrained window with a value of 0.131 V. This observation can be ascribed to the persistent presence of $Li_5B_7S_{13}$—a notably stable phase - in the decomposition products of $Li_4B_7S_{12}Cl$ throughout the examined voltage range. Further analysis, detailed in Appendix C and Table X, suggests that upon reacting with bulk Li metal at $\phi = 0$ V, the reaction energies $\overline{\Delta\Omega}(\phi)$ of thioboracite materials were found to be lower than those of boracite materials, implying that thioboracite materials are comparatively less stable in contact with pure Li metal. A similar trend was observed at a $\phi = 5$ V cathode side, where a comparatively larger driving force for the decomposition of thioboracite materials was noted. However, during the delithiation reaction at high voltages, all boracite materials exhibit the release of gas-phase decompositions of O_2 and/or Cl_2, which may pose significant challenges to the long-term stability of the electrolyte. It is also found that the doping of Al in pure (thio)boracites does not appear to enhance the electrochemical window, except for the case of $Li_4B_7S_{12}Cl$ and $Li_4Al_3B_4S_{12}Cl$, which could be due to the aforementioned reason. The relatively narrow stability windows of Al-doped materials are likely due to the presence of Al-containing compounds in the decomposition products.

When comparing computational results with experimental characterizations, clear discrepancies emerge. for example, the calculated electrochemical stability windows for $Li_4B_7O_{12}Cl$ and $Li_4Al_3B_4O_{12}Cl$ are 2.369 V and 2.271 V, respectively. However, experimental data shows that $Li_4B_7O_{12}Cl$ has a measured electrochemical stability window extending up to 4.3 V [2], while $Li_4Al_3B_4O_{12}Cl$ spans from 0 to 6 V [4]. The lack of agreement can largely be attributed to the passivating effects evidenced in previous studies [43,44,53]. As shown in Table IX, the (thio)boracite materials rarely generate metallic compounds during the reactions (LiB_3 are zero-gap semiconductors [54]). This characteristic implies that these materials may become passivated by forming decomposition products with poor electronic conductivity upon contact with electrodes over an extended voltage range. The formation of passivating products can potentially prevent further degradation of the electrolyte, maintaining optimal electrochemical performance. As a result, the (thio)boracite materials, when synthesized with equilibrium phases, are expected to deliver an electrochemical window that surpasses their inherent intrinsic capacities.

VI. ANALYSIS OF HARMONIC PHONONS OF THE (THIO)BORACITES BASED ON THE IDEALIZED GROUND-STATE STRUCTURES

Omitted from our analysis so far is the consideration of lattice vibrations, which are important for understanding the dynamic stabilities of the materials and for estimating temperature dependent contributions to the Helmholtz free energy. Here we focus on the harmonic approximation, omitting consideration of anharmonic effects. An improvement over Eq. (1) to the estimate of the Helmholtz free energy of a system composed of N_i atoms of type i at temperature T and volume V is given by

$$F(T, V, \{N_i\}) = U_{SL}(\{N_i\}) + F_{\rm vib}(T, V, \{N_i\}). \tag{23}$$

The vibrational contribution to the Helmholtz free energy depends on the frequencies of the normal modes ν of vibration $\omega_\nu(\mathbf{q})$, as a function of phonon wavevector $\mathbf{q}$ sampled throughout the Brillouin zone with weight factors $W(\mathbf{q})$ according to [55]

$$F_{\rm vib}(T, V, \{N_i\}) = k_B T \sum_{\nu\mathbf{q}} W(\mathbf{q}) \left(\ln\left(2\sinh\left(\frac{\hbar\omega_\nu(\mathbf{q})}{2k_B T} \right) \right) \right), \tag{24}$$

where $\hbar$ and k_B denote the Planck and Boltzmann constants, respectively. In order to evaluate Eq. (24) quantitatively as well as to evaluate the phonon band structure and the phonon densities of states, it is necessary to determine the normal mode frequencies $\omega_\nu(\mathbf{q})$ on a reasonably dense $\mathbf{q}$-point grid throughout the Brillouin zone (typically $16 \times 16 \times 16$ sampling points along the three vectors of the reciprocal lattice of the primitive cell). From the pioneering study of Giannozzi, Gonze, and others [10–14], density functional perturbation theory (DFPT) can be used to find the dynamical matrix as a function of $\mathbf{q}$. For computational efficiency, DFPT is used with a relatively coarse $\mathbf{q}$-point sampling ($2 \times 2 \times 2$ or $4 \times 4 \times 4$ sampling points) together with symmetry analysis. These results are then used with Fourier analysis to find the real-space force-constant matrix. This real-space force-constant matrix is then used to find the dynamical matrix and normal mode frequencies $\omega_\nu(\mathbf{q})$ on a $\mathbf{q}$-point grid. This methodology is available in the ABINIT and QUANTUM ESPRESSO packages and in our tests on a few of the systems, we find good agreement between the codes ($\Delta F_{\rm vib}(T, V, \{N_i\}) = \pm 0.01$ eV/FU) when the coarse $\mathbf{q}$-point sampling is well converged. However, we note that variations in the computation of $F_{\rm vib}(T, V, \{N_i\})$ can be much larger (± 0.04 eV/FU for example) when the coarse $\mathbf{q}$-point sampling is not well converged. Another approach, based evaluating the real-space force-constant matrix from finite displacements of the equilibrium atoms within supercells, [56] has been implemented by Togo and coworkers in the PHONOPY code [57,58]. For systems with relatively large primitive cells ($\approx$54 atoms for the Li_6 (thio)boracites), some having low symmetry, we find PHONOPY to be computationally efficient, using QUANTUM ESPRESSO to evaluate the forces in the supercell and also to evaluate the long-range dipolar electric field couplings [10,11,59] using DFPT at $\mathbf{q} = 0$. It is our experience that relatively small supercells ($2 \times 2 \times 2$ multiples of the primitive lattice vectors for the eight Li (thio)boracites and $4 \times 4 \times 4$ multiples of the primitive lattice vectors of Li_2O and Li_2S) give reasonable results. The phonon analysis reported here was generated using this approach.

Using Eq. (24), it is possible to analyze the temperature-dependent energetics of the harmonic phonon contributions to the Helmholtz free energy. Of interest is to measure the magnitude of the vibrational energy, which have been neglected in estimating the equilibrium phase diagrams and in estimating the grand canonical potential analysis. Following the approach of Sec. (III), it is convenient to define free energy differences again based on the Li_4boracite material in the $R3c$ structure. For family members having the Li_4 stoichiometries,

TABLE III. Summary of results for $\Delta F(T, V, \{N_i\})$ in units of eV/formula unit as estimated from Eqs. (25) and (26), referenced to the corresponding Li_4 boracite in its $R3c$ structure plus Li_2O or Li_2S, as appropriate, calculated at temperatures $T = 0$, 300, and 400 K.

Formula	SG	ΔF $T = 0$ K	ΔF $T = 300$ K	ΔF $T = 400$ K
$Li_4B_7O_{12}Cl$	$R3c$	0.00	0.00	0.00
$Li_6B_7O_{13}Cl$	$R3c$	−1.59	−1.53	−1.50
$Li_6B_7O_{13}Cl$	Cc	−1.27	−1.22	−1.20
$Li_4Al_3B_4O_{12}Cl$	$R3c$	0.00	0.00	0.00
$Li_6Al_3B_4O_{13}Cl$	$R3c$	−0.25	−0.20	−0.18
$Li_6Al_3B_4O_{13}Cl$	Cc	−0.42	−0.35	−0.32
$Li_4B_7S_{12}Cl$	$R3c$	0.00	0.00	0.00
$Li_4B_7S_{12}Cl$	$F\bar{4}3c$	−0.07	−0.09	−0.10
$Li_6B_7S_{13}Cl$	$R3c$	−0.96	−0.92	−0.89
$Li_6B_7S_{13}Cl$	Cc	−1.05	−0.99	−0.96
$Li_4Al_3B_4S_{12}Cl$	$R3c$	0.00	0.00	0.00
$Li_6Al_3B_4S_{13}Cl$	$R3c$	−0.33	−0.23	−0.19
$Li_6Al_3B_4S_{13}Cl$	Cc	−0.63	−0.52	−0.47

$\Delta F(T, \text{Li}_4\text{boracite})$ is defined as

$$\Delta F(T, \text{Li}_4\text{boracite}) \equiv F(T, \text{Li}_4\text{boracite})) - F(T, \text{Li}_4\text{boracite}[R3c]). \quad (25)$$

For family members having the Li_6 stoichiometries, $\Delta F(T, \text{Li}_6\text{boracite})$ is defined as

$$\Delta F(T, \text{Li}_6\text{boracite}) \equiv F(T, \text{Li}_6\text{boracite}) - F(T, \text{Li}_4\text{boracite}[R3c]) - F(T, \text{Li}_2\text{X}). \quad (26)$$

Here Li_2X denotes Li_2O or Li_2S as appropriate. Table III lists $\Delta F(T, V, \{N_i\})$ accordingly, calculated at temperatures $T = 0$, 300, and 400 K. Here we see that the vibrational contributions to the Helmholtz free energy contributions are less than 0.03 eV/FU at $T = 0$ K and less than 0.08 eV/FU at $T = 300$ K, small enough to predict the same phase stability patterns as the static lattice energies given in Table I. Of the 16 materials listed in Table I, only the 13 materials with negative static lattice energies ΔU_{SL} are included in Table III (omitting three of the Li_4 materials with the ordered $F\bar{4}3c$ structures).

In performing the analysis of the normal mode spectra of the 13 materials, we find that, with one exception, all are dynamically stable by the criterion of finding no imaginary modes. The exception is $Li_6B_7S_{13}Cl$ in the $R3c$ structure, which is found to have some imaginary modes in a small volume of the Brillouin zone near $\mathbf{q} = 0$. The volume of phase space, which contributes to these imaginary modes is so small as to have a negligible numerical contribution to the phonon density of states, but nevertheless indicates some dynamic instability of this structure. The normal mode band diagrams together with partial densities of states profiles of all 13 materials are given in the of the Part I of the SM [36], while those of the eight most stable structures are presented here. Here, to represent the phonon band dispersions of the normal mode frequencies $\omega_\nu(\mathbf{q})$, we use the "standard" Brillouin zone paths for the $F\bar{4}3c$, $R3c$, and Cc structures established by

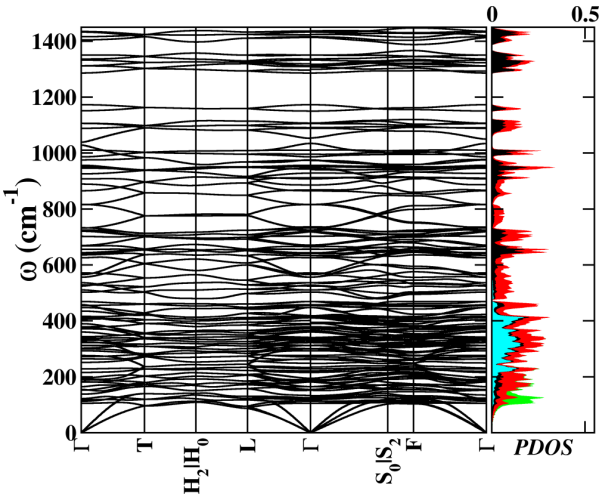


FIG. 11. Phonon bands and densities of states for $Li_4B_7O_{12}Cl$ in the $R3c$ structure.

Hinusa *et al.* [60], which are reproduced in the Appendix D in Fig. 19. Additionally, the partial densities of states $g_a(\omega)$, calculated from the normal mode eigenstates for each atom type a as given, for example, in Eq. (13) of Ref. [22], are also visualized. In this paper, $g_a(\omega)$ is calculated by representing the δ function as a normalized Gaussian function with a smearing width of 1.7 cm^{-1} as calculated within the PHONOPY code. The total density of phonon states $g(\omega)$ is given as the sum of the partial density of states contributions,

$$g(\omega) = \sum_a g_a(\omega). \quad (27)$$

In the following figures, the phonon frequencies are given in units of cm^{-1}. The side panel presents the phonon densities of states $g(\omega)$ in units of (states per unit cell)/cm^{-1} as indicated with the full curve. Within the full curve, the

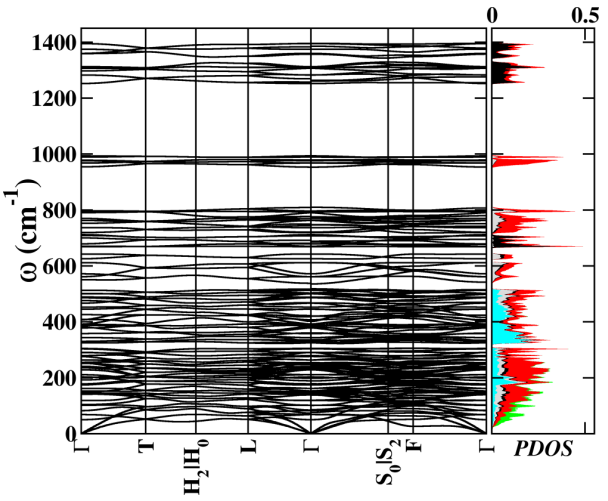


FIG. 12. Phonon bands and densities of states for $Li_4Al_3B_4O_{12}Cl$ in the $R3c$ structure.

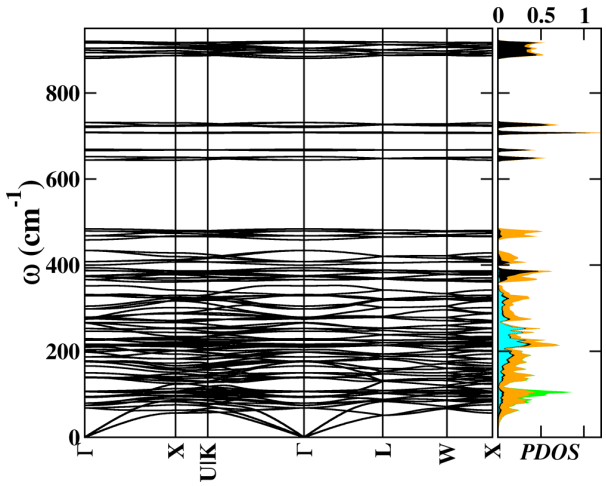


FIG. 13. Phonon bands and densities of states for $Li_4B_7S_{12}Cl$ in the $F\bar{4}3c$ structure.

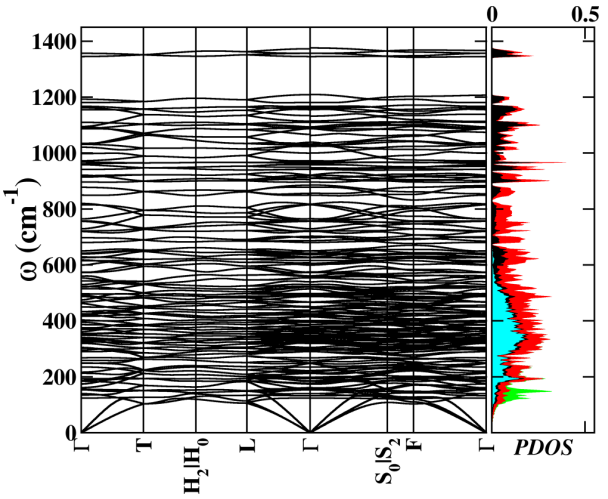


FIG. 15. Phonon bands and densities of states for $Li_6B_7O_{13}Cl$ in the $R3c$ structure.

partial density of states components $g_a(\omega)$ are represented by shaded segments; the width at each ω indicating the magnitude of $g_a(\omega)$. The curve segments are presented in the following order: Li (light blue), Al (grey, if present), B (black), O (red) or S (orange), and Cl (green). The phonon spectrum of the most stable Li (thio)boracites are presented in Figs. 11–18, representing $Li_4B_7O_{12}Cl$ in the $R3c$ structure, $Li_4Al_3B_4O_{12}Cl$ in the $R3c$ structure, $Li_4B_7S_{12}Cl$ in the $F\bar{4}3c$ structure, $Li_4Al_3B_4S_{12}Cl$ in the $R3c$ structure, $Li_6B_7O_{13}Cl$ in the $R3c$ structure, and $Li_6Al_3B_4O_{13}Cl$, $Li_6B_7S_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$ all in the Cc structure, respectively. The phonon band diagrams and phonon partial density of states for all of the 13 Li (thio)boracites listed in Table III are given in Sec. III of Part I of the SM [36]. (Note that in this section of the manuscript and in the corresponding section of the SM, the label "PDOS" refers to the phonon partial density of states.)

These phonon band diagrams have some interesting features. For example, the Li contributions are generally confined to the low-frequency range of $0 \leqslant \omega \leqslant 500 - 600$ cm^{-1}. As the most massive ion in the Li (thio)boracite family, the Cl vibrational contributions are confined to small frequencies less than 200 cm^{-1}. Both O and S contribute throughout the phonon spectrum, participating both in modes associated with Li and Cl ions in the void regions and in the framework modes due to localized vibrations of the BO_4, AlO_4, BS_4, and AlS_4 tetrahedra and the BO_3 and BS_3 triangular framework components. The triangular planar structures are responsible for the highest frequency modes for all of the Li boracite materials and for the Li_4 thioboracites. Since Al is more massive than B, it is not surprising that the framework modes associated with the AlO_4 and AlS_4 tetrahedral units generally occur at lower frequencies than those of the BO_4 and BS_4 tetrahedral units. It is interesting to note that of the 13 stable and metastable

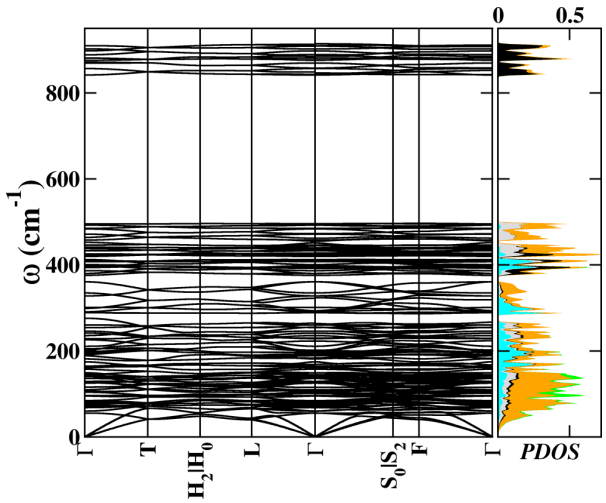


FIG. 14. Phonon bands and densities of states for $Li_4Al_3B_4S_{12}Cl$ in the $R3c$ structure.

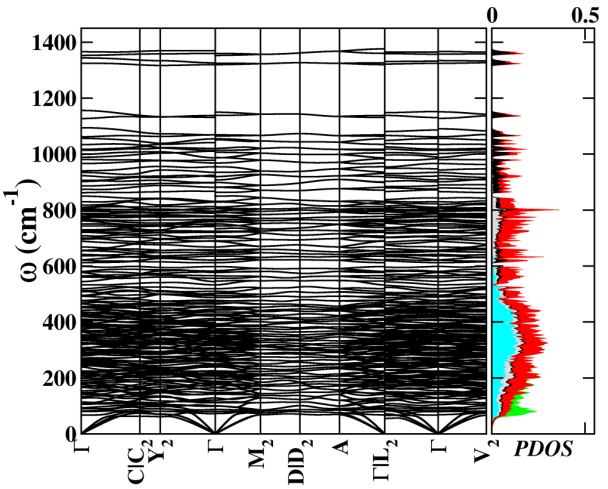


FIG. 16. Phonon bands and densities of states for $Li_6Al_3B_4O_{12}Cl$ in the Cc structure.

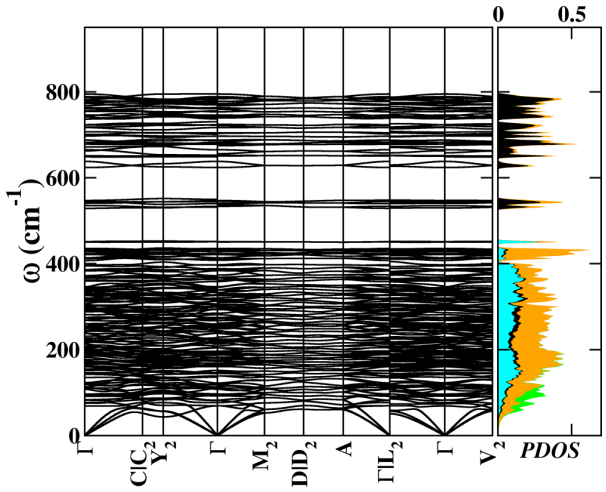


FIG. 17. Phonon bands and densities of states for $Li_6B_7S_{12}Cl$ in the *Cc* structure.

structures considered in this paper, only $Li_6B_7S_{13}Cl$ in the *R*3*c* crystalline form shows evidence of imaginary phonon frequencies. As shown in Fig. 8 of Part I of the SM [36], these imaginary modes occur in a small volume of the Brillouin zone near $\mathbf{q} = 0$. Interestingly, as discussed in Sec. III, this material shows considerable similarity to $Li_6B_7S_{13}I$ found experimentally by Kaup and coworkers [6] in the disordered $F\bar{4}3c$ structure. It is possible that the halide iodine contributes some dynamic stabilization compared with chlorine or that the simulations are missing significant properties of the real materials [61]. On the other hand, the related monoclinic structure (*Cc*; No. 9) of $Li_6B_7S_{13}Cl$ and of several of the other Li_6 (thio)boracites is computational predicted to be more stable by 0.1 eV/FU or more and are all predicted to be dynamically stable. To our knowledge, these *Cc* structures have not yet been reported in the experimental literature.

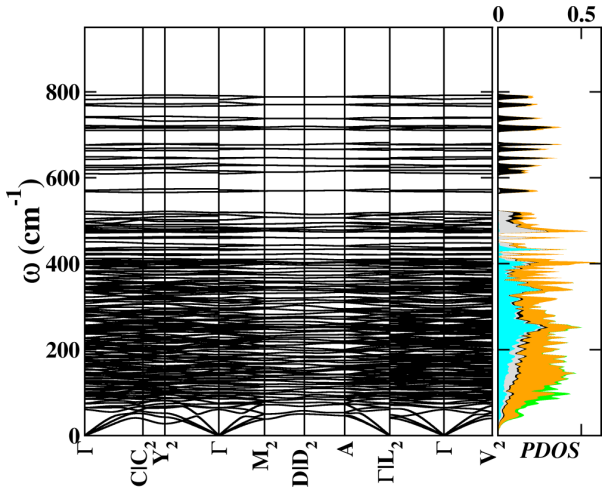


FIG. 18. Phonon bands and densities of states for $Li_6Al_3B_4S_{12}Cl$ in the *Cc* structure.

VII. SUMMARY AND CONCLUSIONS

In this paper we have built upon previous computational and experimental investigations regarding known materials ($Li_4B_7O_{12}Cl$, $Li_4Al_3B_4O_{12}Cl$, and $Li_6B_7S_{13}I$) and extended this family of (thio)boracites to include newly predicted materials based on our computational efforts ($Li_4B_7S_{12}Cl$, $Li_4Al_3B_4S_{12}Cl$, $Li_6B_7O_{13}Cl$, $Li_6Al_3B_4O_{13}Cl$, $Li_6B_7S_{13}Cl$, and $Li_6Al_3B_4S_{13}Cl$).

We have discovered new structures apart from the previously known ordered *R*3*c* and disordered $F\bar{4}3c$ phases, namely new ordered *Cc* and $F\bar{4}3c$ structures. For the Li_4 family of materials we have investigated, we have determined that the energetically favorable ground states for all materials prefer the *R*3*c* structure, except the computationally predicted $Li_4B_7S_{12}Cl$, which has a lowest-energy state in the $F\bar{4}3c$ structure, with fully occupied Li 32e sites in the conventional cell. For the Li_6 materials, we have found a different pattern, that they all energetically favor the *Cc* structure, except for the computationally predicted $Li_6B_7O_{13}Cl$, which has a lowest-energy structure with *R*3*c* symmetry.

We have adopted several methodologies for investigating the stability of these materials, including convex hull analysis of the composition spaces of Li-Al-B-O-Cl and Li-Al-B-S-Cl, estimation of voltage windows in practical applications, and phonon mode analysis.

The convex hull analysis suggests that the $Li_4B_7O_{12}Cl$ material is the only one that is chemically stable, but this is not quite the full story. Materials with very small positive values of hull energy may be stable, or at the very least metastable. For example, the experimentally synthesized and investigated $Li_4Al_3B_4O_{12}Cl$ is computed to have the small positive hull energy of 0.053 eV/atom. This small unexpected positive energy may be caused by the limited accuracy of density functional calculations or by the simplifying approximations to the physical system made in the analysis. The sensitivity of the results to the choice of the exchange-correlation functional may be worth examining in future work. While the PBEsol functional [16] has been successfully used in our previous work [22] and by many other authors [62], new functionals appearing in the more recent literature may possibly predict a slightly different phase diagram.

The voltage window analysis attempts to model real world stability of these materials when placed in contact with a pure Li metal anode and with a voltage applied across the solid electrolyte. The results from this model suggest that all of the oxygen-based materials have good stability in contact with Li metal, whereas the sulfur-based materials do not. The low stability in contact with Li metal found here does not preclude the thioboracites from being useful in practice, as this may be addressable with anode coatings and/or by using another anode that is not pure Li.

From the phonon calculations, we have generated the vibrational energies at 0 K, 300 K, and 400 K, and these data points further solidify our previous ground-state predictions from the static lattice energies alone. The phonon modes themselves also indicate vibrational stability of our ground-state structures, as they all have real frequencies. The only structure investigated that was found to have instability as indicated by imaginary phonon modes was $Li_6B_7S_{13}Cl$ in its

TABLE IV. Conventional cell fractional coordinates of $Li_4B_7O_{12}Cl$ from the optimized structure (face-centered $R3c$) compared with the experimental structure at room temperature (face-centered-cubic $F\bar{4}3c$) reported in Ref. [5]. The corresponding lattice parameters are given in Table I. The "Label" columns with notation $8/32e$ give the (number of occupied sites)/(site multiplicity)(Wyckoff label) for the appropriate space group. The multiplicity for each listed site includes the full face-centered translations of (0, 0, 0), $(\frac{1}{2}, \frac{1}{2}, 0)$, $(0, \frac{1}{2}, \frac{1}{2})$, and $(\frac{1}{2}, 0, \frac{1}{2})$.

Atom	Label	(x, y, z) for $R3c$	Label	(x, y, z) for $F\bar{4}3c$
Li	24/24*b*	(0.035,0.251,0.250)	24/24*c*	(0.000,0.250,0.250)
Li	8/8*a*	(0.369,0.369,0.369)	8/32*e*	(0.361,0.361,0.361)
B	24/24*b*	(0.256,0.007,0.002)	24/24*d*	(0.250,0.000,0.000)
B	8/8*a*	(0.102,0.102,0.102)	32/32*e*	(0.101,0.101,0.101)
B	24/24*b*	(0.903,0.903,0.109)		(0.899,0.899,0.101)
O	24/24*b*	(0.026,0.097,0.185)	96/96*h*	(0.022,0.099,0.182)
O	24/24*b*	(0.979,0.901,0.193)		(0.978,0.901,0.182)
O	24/24*b*	(0.981,0.106,0.822)		(0.978,0.099,0.818)
O	24/24*b*	(0.024,0.909,0.827)		(0.022,0.901,0.818)
Cl	8/8*a*	(0.250,0.250,0.250)	8/8*b*	(0.250,0.250,0.250)

$R3c$ structure, while in its ground-state Cc structure, all of the phonon modes are real.

The predicted materials investigated in this paper seem quite promising and it seems plausible that there are viable routes of synthesis for these materials and that they may be metastable with respect to their precursors. This paper has established the structures and stability of this very intriguing family of (thio)boracites, but more investigation is needed to determine other properties. Future work is planned to investigate more of these properties, such as ionic conductivity, electrolyte performance, and ion migration mechanisms.

ACKNOWLEDGMENTS

The work by D.C.L., Y.L., and N.A.W.H. was supported by NSF Grants No. DMR-1940324 and No. DMR-2242959; the work by Y.L. and P.C. was supported by the National Research Foundation, Singapore under the NRF Fellowship NRFF12-2020-0012. Computations were performed using the Wake Forest University (WFU) High Performance Computing Facility, a centrally managed computational resource available to WFU researchers including faculty, staff, students, and collaborators. We would like to specially thank Dr. Sean Anderson for help with the software on the WFU-HPC computer system. We also like to thank Prof. Stephen Winter and Dr. Saif Ullah at WFU for their helpful comments.

APPENDIX A: TABLES OF FRACTIONAL COORDINATES

In this Appendix we list the computed fractional coordinates of the optimized structures described in Sec. III of this paper, compared with experimental results from the literature when available. The listed coordinates are chosen among their equivalents so that the relationships between the materials can be compared, but of course are not unique. The full primitive cell coordinates from which these results are provided in the Part I of the SM [36].

In Table IV we list the computed $R3c$ fractional coordinates of the optimized structure of $Li_4B_7O_{12}Cl$ slightly updated from the previously reported in Ref. [3] and compared with

TABLE V. Conventional cell fractional coordinates of $Li_4Al_3B_4O_{12}Cl$ from the optimized structure (face-centered $R3c$) compared with the experimental structure at room temperature (face-centered-cubic $F\bar{4}3c$) reported in Ref. [5]. The notation is the same as that in Table IV.

Atom	Label	(x, y, z) for $R3c$	Label	(x, y, z) for $F\bar{4}3c$
Li	24/24*b*	(0.963,0.223,0.223)	24/24*c*	(0.000,0.250,0.250)
Li	8/8*a*	(0.354,0.354,0.354)	8/32*e*	(0.364,0.364,0.364)
Al	24/24*b*	(0.225,0.980,0.979)	24/24*d*	(0.250,0.000,0.000)
B	8/8*a*	(0.086,0.086,0.086)	32/32*e*	(0.106,0.106,0.106)
B	24/24*b*	(0.871,0.871,0.081)		(0.894,0.894,0.106)
O	24/24*b*	(0.008,0.091,0.155)	96/96*h*	(0.027,0.109,0.178)
O	24/24*b*	(0.950,0.865,0.152)		(0.973,0.891,0.178)
O	24/24*b*	(0.947,0.089,0.800)		(0.973,0.109,0.822)
O	24/24*b*	(0.996,0.870,0.804)		(0.027,0.891,0.822)
Cl	8/8*a*	(0.250,0.250,0.250)	8/8*b*	(0.250,0.250,0.250)

TABLE VI. Conventional cell fractional coordinates of $Li_6B_7S_{13}Cl$ from the optimized structure (face-centered $R3c$) compared with the corresponding coordinates of $Li_6B_7S_{13}I$ reported in the supplemental materials of Ref. [6], shifted by $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, for the $F\bar{4}3c$ structure. The notations is the same that given in IV.

Atom	Label	$R3c$ (x, y, z)	Label	$F\bar{4}3c$ (x, y, z)
Li	24/24*b*	(0.087,0.250,0.252)	16/24*c*	(0.000,0.250,0.250)
Li	24/24*b*	(0.889,0.880,0.163)	32/32*e*	(0.859,0.859,0.141)
B	24/24*b*	(0.271,0.021,0.018)	24/24*d*	(0.250,0.000,0.000)
B	8/8*a*	(0.443,0.443,0.443)	32/32*e*	(0.423,0.423,0.423)
B	24/24*b*	(0.598,0.596,0.443)		(0.577,0.577,0.423)
S	24/24*b*	(0.038,0.202,0.412)	96*h*	(0.014,0.181,0.395)
S	24/24*b*	(0.007,0.839,0.415)		(0.986,0.819,0.395)
S	24/24*b*	(0.031,0.840,0.626)		(0.014,0.819,0.605)
S	24/24*b*	(0.005,0.201,0.623)		(0.986,0.181,0.605)
S	8/8*a*	(0.020,0.020,0.020)	8/8*a*	(0.000,0.000,0.000)
Cl/I	8/8*a*	(0.250,0.250,0.250)	8/8*b*	(0.250,0.250,0.250)

the recent literature results of Ref. [5] in the basis of the conventional face-centered unit cells. In Table V the computed optimized $R3c$ fractional coordinates of $Li_4Al_3B_4O_{12}Cl$ are listed and compared with the recent literature results of Ref. [5] in the basis of the conventional face-centered unit cells. In Table VI the computed optimized $R3c$ fractional coordinates of $Li_6B_7S_{13}Cl$ are listed and compared with the literature results for cubic $Li_6B_7S_{13}I$ reported in Ref. [6] in the basis of the conventional face-centered unit cells. In Table VII the computed optimized $R3c$ fractional coordinates of $Li_6B_7O_{13}Cl$ and $Li_6Al_3B_4O_{13}Cl$ are listed in the basis of the conventional face-centered unit cells.

APPENDIX B: FORMATION ENERGIES AND CONVEX HULL ANALYSIS FOR MATERIALS RELATED TO THE LI (THIO)BORACITE FAMILY

This appendix presents in Table VIII the essential data used to generate the phase diagrams in Figs. 8 and 9. This includes the reduced formula form of each material in column 2. In cases where a material has multiple phases, only the lowest-energy phases are included in the table, with symmetry information (space group symbol and number in the bracket [30]) given in column 3. Detailed structural information for these lowest-energy structures are available in the Part II of the SM [36].

APPENDIX C: ENERGY TABLES RELATED TO THE ANALYSIS OF THE VOLTAGE STABILITY OF LI (THIO)BORACITE FAMILY

Tables IX and X list some of the details of the analysis of the grand canonical potential for the Li (thio)boracites.

APPENDIX D: "STANDARD" BRILLOUIN ZONE PATHS AND LABELS

For convenience we present in Fig. 19 the recommended Brillouin zone labels and paths according to Ref. [60] used in the phonon band diagrams of this paper. This choice is an update of the previous study of Setyawan and Curtarola [65].

TABLE VII. Conventional cell-fractional coordinates (x, y, z) of $Li_6B_7O_{13}Cl$ from the optimized structure (face-centered $R3c$) compared with the corresponding coordinates for $Li_6Al_3B_4O_{13}Cl$. The notations is the same that given in Table IV.

Atom	Label	$Li_6B_7O_{13}Cl$	$Li_6Al_3B_4O_{13}Cl$
Li	24/24*b*	(0.036,0.241,0.238)	(0.039,0.238,0.245)
Li	24/24*b*	(0.875,0.870,0.139)	(0.889,0.872,0.141)
B	24/24*b*	(0.255,0.003,0.004)	(0.270,0.015,0.005)
B	8/8*a*	(0.403,0.403,0.403)	(0.421,0.421,0.421)
B/Al	24/24*b*	(0.585,0.584,0.430)	(0.601,0.592,0.431)
O	24/24*b*	(0.096,0.019,0.329)	(0.924,0.994,0.342)
O	24/24*b*	(0.024,0.185,0.408)	(0.114,0.013,0.323)
O	24/24*b*	(0.982,0.898,0.821)	(0.991,0.217,0.605)
O	24/24*b*	(0.978,0.191,0.603)	(0.913,0.023,0.704)
O	8/8*a*	(0.016,0.016,0.016)	(0.023,0.023,0.023)
Cl	8/8*a*	(0.250,0.250,0.250)	(0.250,0.250,0.250)

TABLE VIII. Results for compounds in the Li-B-Al-O-Cl and Li-B-Al-S-Cl systems assessed in this study. The "Formation" and "Hull" columns provide the formation energy [calculated based on Eq. (7)] and the energy above the hull [calculated using Eq. (8) to Eq. (10)], respectively, for each compound listed in the second column. The "Equilibrium" column specifies the equilibrium phases with their corresponding stoichiometric coefficients. All energy values are presented in eV/atom. Note that entry #65 for $Li_{7.5}B_{10}S_{18}Cl_{1.5}$ was based on the material reported in Ref. [63], using the crystal structure described in Ref. [64].

No.	Material	Symmetry	Formation	Hull	Equilibrium
1	Li	$Im\bar{3}m(229)$	0.000	0.000	1.00 Li
2	Al	$F3m3(225)$	0.000	0.000	1.00 Al
3	B	$R\bar{3}m(166)$	0.000	0.000	1.00 B
4	S	$P2/c(13)$	0.000	0.000	1.00 S
5	O ($\frac{1}{2}$ O_2)	$C2/m(12)$	0.000	0.000	1.00 O
6	Cl ($\frac{1}{2}$ Cl_2)	$Cmce(64)$	0.000	0.000	1.00 Cl
7	Li_9Al_4	$C2/m(12)$	−0.111	0.036	4.00 Li_2Al + 1.00 Li
8	LiAl	$I4_1/amd(141)$	−0.176	0.000	1.00 LiAl
9	$LiAl_3$	$Pm\bar{3}m(221)$	−0.089	0.000	1.00 $LiAl_3$
10	Li_2Al	$Cmcm(63)$	−0.159	0.000	1.00 Li_2Al
11	Li_3Al_2	$R\bar{3}m(166)$	−0.123	0.043	1.00 Li_2Al + 1.00 LiAl
12	LiB_3	$P4/mbm(127)$	−0.216	0.000	1.00 LiB_3
13	Li_5B_4	$Cm(8)$	0.150	0.278	1.33 LiB_3 + 3.67 Li
14	LiB	$P6_3/mmc(194)$	−0.124	0.021	0.33 LiB_3 + 0.67 Li
15	Li_2S	$Fm\bar{3}m(225)$	−1.366	0.000	1.00 Li_2S
16	Li_2O	$Fm\bar{3}m(225)$	−1.842	0.000	1.00 Li_2O
17	Li_2O_2	$P6_3/mmc(194)$	−1.416	0.000	1.00 Li_2O_2
18	LiCl	$Fm\bar{3}m(225)$	−1.843	0.000	1.00 LiCl
19	AlB_2	$P6/mmm(191)$	−0.072	0.000	1.00 AlB_2
20	AlB_{12}	$P4_12_12(92)$	0.336	0.353	1.00 AlB_2 + 10.00 B
21	Al_2S_3	$I4_1/amd(141)$	−1.121	0.000	1.00 Al_2S_3
22	Al_2O_3	$R\bar{3}c(167)$	−3.062	0.000	1.00 Al_2O_3
23	$AlCl_3$	$C2/m(12)$	−1.549	0.000	1.00 $AlCl_3$
24	BS	$R\bar{3}m(166)$	−0.452	0.000	1.00 BS
25	BS_2	$P2_1/c(14)$	−0.367	0.000	1.00 BS_2
26	B_2S_3	$P2_1/c(14)$	−0.395	0.006	1.00 BS_2 + 1.00 BS
27	B_6O	$R\bar{3}m(166)$	−0.715	0.000	1.00 B_6O
28	B_2O_3	$Cmc2_1(36)$	−2.414	0.000	1.00 B_2O_3
29	BCl_2	$Pbca(61)$	−0.701	0.127	0.50 BCl + 0.50 BCl_3
30	BCl	$Pa\bar{3}(205)$	−0.629	0.000	1.00 BCl
31	BCl_3	$P6_3/m(176)$	−0.928	0.000	1.00 BCl_3
32	SCl_2	$P2_12_12_1(19)$	−0.102	0.000	1.00 SCl_2
33	SCl	$Fdd2(43)$	−0.148	0.000	1.00 SCl
34	Cl_2O	$I4_1/amd(141)$	0.249	0.249	0.50 O_2 + 1.00 Cl_2
35	ClO_3	$Cc(9)$	0.133	0.133	1.50 O_2 + 0.50 Cl_2
36	Cl_2O_7	$C2/c(15)$	0.168	0.168	3.50 O_2 + 1.00 Cl_2
37	ClO_2	$Pbca(61)$	0.104	0.104	1.00 O_2 + 0.50 Cl_2
38	$LiAlB_{14}$	$C2(5)$	0.113	0.181	1.00 AlB_2 + 1.00 LiB_3 + 9.00 B
39	Li_5AlS_4	$P2_1/m(11)$	−1.339	0.000	1.00 Li_5AlS_4
40	Li_5AlO_4	$Pbca(61)$	−2.219	0.000	1.00 Li_5AlO_4
41	$LiAlO_2$	$R\bar{3}m(166)$	−2.761	0.000	1.00 $LiAlO_2$
42	$LiAl_5O_8$	$P4_332(212)$	−2.984	0.000	1.00 $LiAl_5O_8$
43	$LiAlCl_4$	$P2_1/c(14)$	−1.639	0.009	1.00 $AlCl_3$ + 1.00 LiCl
44	$Li_2B_2S_5$	$Cmcm(63)$	0.156	0.914	0.29 $Li_5B_7S_{13}$ + 0.29 Li_2S + 1.00 S
45	$Li_5B_7S_{13}$	$Cc(9)$	−0.791	0.000	1.00 $Li_5B_7S_{13}$

TABLE VIII. (*Continued.*)

No.	Material	Symmetry	Formation	Hull	Equilibrium
46	Li_3BS_3	$Pnma(62)$	−1.044	0.029	0.14 $Li_5B_7S_{13}$ + 1.14 Li_2S
47	$Li_2B_4O_7$	$I4_1cd(110)$	−2.445	0.002	0.50 $LiBO_2$ + 0.50 $Li_3B_7O_{12}$
48	$Li_3B_{11}O_{18}$	$P2_1/c(14)$	−2.427	0.011	1.00 $Li_3B_7O_{12}$ + 2.00 B_2O_3
49	LiB_3O_5	$Pna2_1(33)$	−2.435	0.007	0.33 $Li_3B_7O_{12}$ + 0.33 B_2O_3
50	Li_3BO_3	$P2_1/c(14)$	−2.262	0.000	1.00 Li_3BO_3
51	$Li_3B_7O_{12}$	$P\bar{1}(2)$	−2.449	0.000	1.00 $Li_3B_7O_{12}$
52	$Li_6B_4O_9$	$P2_1/c(14)$	−2.339	0.030	1.00 Li_3BO_3+ 3.00 $LiBO_2$
53	$LiBO_2$	$I\bar{4}2d(122)$	−2.432	0.000	1.00 $LiBO_2$
54	$LiClO_2$	$P4_2/ncm(138)$	−0.747	0.175	1.00 LiCl + 1.00 O_2
55	Li_5Cl_3O	$Fmm2(42)$	−1.617	0.226	1.00 Li_2O + 3.00 LiCl
56	$LiClO_4$	$Pnma(62)$	−0.613	0.002	1.00 LiCl + 2.00 O_2
57	Al_5BO_9	$Cmc2_1(36)$	−2.951	0.004	0.17 $Al_4B_6O_{15}$ + 2.17 Al_2O_3
58	$Al_4B_6O_{15}$	$R3(146)$	−2.680	0.000	1.00 $Al_4B_6O_{15}$
59	$AlSCl_7$	$Pc(7)$	−0.818	0.000	1.00 $AlSCl_7$
60	AlClO	$Pmmn(59)$	−2.389	0.0004	0.33 $AlCl_3$ + 0.33 Al_2O_3
61	Li_2AlBO_4	$P2_1/c(14)$	−2.549	0.048	1.00 $LiBO_2$ + 1.00 $LiAlO_2$
62	$Li_2AlB_5O_{10}$	$P2_1/c(14)$	−2.513	0.021	0.06 $Al_4B_6O_{15}$ + 0.67 $Li_3B_7O_{12}$ + 0.39 Al_2O_3
63	$LiAlB_2O_5$	$C2/c(15)$	−2.493	0.124	0.25 $LiBO_2$ + 0.25 $Li_3B_7O_{12}$ + 0.50 Al_2O_3
64	$Li_3AlB_2O_6$	$P\bar{1}(2)$	−2.490	0.052	2.00 $LiBO_2$ + 1.00 $LiAlO_2$
65	$Li_{7.5}B_{10}S_{18}Cl_{1.5}$	$C2/c(15)$	−0.840	0.000	1.00 $Li_{7.5}B_{10}S_{18}Cl_{1.5}$
66	$Li_5B_7O_{12.5}Cl$	$F23(196)$	−2.413	0.000	1.00 $Li_5B_7O_{12.5}Cl$
67	$Li_4B_7O_{12}Cl$	$R3c(161)$	−2.406	0.000	1.00 $Li_4B_7O_{12}Cl$
68	$Li_4B_7O_{12}Cl$	$F\bar{4}3c(219)$	−2.397	0.009	1.00 $Li_4B_7O_{12}Cl$
69	$Li_6B_7O_{13}Cl$	$R3c(161)$	−2.402	0.004	2.33 $LiBO_2$ + 0.66 $Li_5B_7O_{12.5}Cl$ + 0.33 LiCl
70	$Li_6B_7O_{13}Cl$	$Cc(9)$	−2.390	0.016	2.33 $LiBO_2$ + 0.66 $Li_5B_7O_{12.5}Cl$ + 0.33 LiCl
71	$Li_4Al_3B_4O_{12}Cl$	$R3c(161)$	−2.539	0.053	0.27 $LiBO_2$ + 0.60 $LiAl_5O_8$ + 0.54 $Li_5B_7O_{12.5}Cl$ + 0.47 LiCl
72	$Li_4Al_3B_4O_{12}Cl$	$F\bar{4}3c(219)$	−2.489	0.103	0.27 $LiBO_2$ + 0.60 $LiAl_5O_8$ + 0.54 $Li_5B_7O_{12.5}Cl$ + 0.47 LiCl
73	$Li_6Al_3B_4O_{13}Cl$	$R3c(161)$	−2.469	0.087	4.00 $LiBO_2$ + 0.50 $LiAl_5O_8$ + 0.50 $LiAlO_2$ + 1.00 LiCl
74	$Li_6Al_3B_4O_{13}Cl$	$Cc(9)$	−2.476	0.080	4.00 $LiBO_2$ + 0.50 $LiAl_5O_8$ + 0.50 $LiAlO_2$ + 1.00 LiCl
75	$Li_4B_7S_{12}Cl$	$R3c(161)$	−0.754	0.015	0.50 $Li_{7.5}B_{10}S_{18}Cl_{1.5}$ + 1.00 BS_2 + 1.00 BS + 0.25 LiCl
76	$Li_4B_7S_{12}Cl$	$F\bar{4}3c(219)$	−0.757	0.013	0.50 $Li_{7.5}B_{10}S_{18}Cl_{1.5}$ + 1.00 BS_2 + 1.00 BS + 0.25 LiCl
77	$Li_6B_7S_{13}Cl$	$R3c(161)$	−0.858	0.011	1.00 $Li_5B_7S_{13}$ + 1.00 LiCl
78	$Li_6B_7S_{13}Cl$	$Cc(9)$	−0.861	0.008	1.00 $Li_5B_7S_{13}$ + 1.00 LiCl
79	$Li_4Al_3B_4S_{12}Cl$	$R3c(161)$	−0.928	0.059	0.57 $Li_5B_7S_{13}$ + 0.03 Li_5AlS_4 + 1.49 Al_2S_3 + 1.00 LiCl
80	$Li_4Al_3B_4S_{12}Cl$	$F\bar{4}3c(219)$	−0.893	0.095	0.57 $Li_5B_7S_{13}$ + 0.03 Li_5AlS_4 + 1.49 Al_2S_3 + 1.00 LiCl
81	$Li_6Al_3B_4S_{13}Cl$	$R3c(161)$	−0.990	0.045	0.57 $Li_5B_7S_{13}$ + 0.43 Li_5AlS_4 + 1.29 Al_2S_3 + 1.00 LiCl
82	$Li_6Al_3B_4S_{13}Cl$	$Cc(9)$	−1.001	0.034	0.57 $Li_5B_7S_{13}$ + 0.43 Li_5AlS_4 + 1.29 Al_2S_3 + 1.00 LiCl

TABLE IX. Decomposition equations and corresponding reaction energies at critical applied voltages or chemical potentials. Results are listed in eight subtables for the each of the Li (thio)boracites in this study based on their most stable structural form. In each subtable, the first column gives the fixed value of $\phi = -\mu_{Li}(\phi)/e$ in units of Volts. The second column presents the computed values of $\overline{\Delta\Omega}$ in units of eV/atom, defined by Eq. (22) and derived from Eq. (21) corresponding to the most energetically favorable reaction, which is listed in the final column. That reaction is found using the Pymatgen software [39] according to the lowest energy of $\Delta\Omega$(system, ϕ, Δn_{Li}) found for the given electrolyte (R) and voltage (ϕ) according to Eq. (20). It should be noted that the stoichiometric coefficients presented each reaction equation have been rounded to three decimal places. As explained in Sec. IV, the formation energies of elemental Li, Al, B, or S and molecular dimers $\frac{1}{2}$ O_2 or $\frac{1}{2}$ Cl_2 are each zero.

(a) $Li_4B_7O_{12}Cl$

ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−0.714	$Li_4B_7O_{12}Cl$ + 23.333 Li → 12 Li_2O + 2.333 LiB_3 + LiCl
0.311	−0.351	$Li_4B_7O_{12}Cl$ + 10 Li → 4 Li_3BO_3 + LiB_3 + LiCl
0.661	−0.176	$Li_4B_7O_{12}Cl$ + 8.471 Li → 3.824 Li_3BO_3 + 0.529 B_6O + LiCl
0.860	−0.092	$Li_4B_7O_{12}Cl$ + 2.909 Li → 5.909 $LiBO_2$ + 0.182 B_6O + LiCl
1.491	0.000	$Li_4B_7O_{12}Cl$ → $Li_4B_7O_{12}Cl$
3.860	0.000	$Li_4B_7O_{12}Cl$ → $Li_3B_7O_{12}$ + 0.5 Cl_2 + Li
3.881	−0.001	$Li_4B_7O_{12}Cl$ → 3.5 B_2O_3 + 0.75 O_2 + 0.5 Cl_2 + 4 Li
5.000	−0.225	$Li_4B_7O_{12}Cl$ → 3.5 B_2O_3 + 0.75 O_2 + 0.5 Cl_2 + 4 Li

TABLE IX. (*Continued.*)

(b) $Li_6B_7O_{13}Cl$		
ϕ (V) (eV)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−0.600	$Li_6B_7O_{13}Cl + 23.33\ Li \rightarrow 13\ Li_2O + 2.333\ LiB_3 + LiCl$
0.311	−0.254	$Li_6B_7O_{13}Cl + 8.889\ Li \rightarrow 4.333\ Li_3BO_3 + 0.889\ LiB_3 + LiCl$
0.661	−0.106	$Li_6B_7O_{13}Cl + 7.529\ Li \rightarrow 4.176\ Li_3BO_3 + 0.471\ B_6O + LiCl$
0.860	−0.034	$Li_6B_7O_{13}Cl + 1.455\ Li \rightarrow 6.455\ LiBO_2 + 0.091\ B_6O + LiCl$
1.357	0.000	$Li_6B_7O_{13}Cl \rightarrow Li_6B_7O_{13}Cl$
3.607	0.000	$Li_6B_7O_{13}Cl \rightarrow Li_4B_7O_{12}Cl + 0.5\ O_2 + 2\ Li$
3.860	−0.024	$Li_6B_7O_{13}Cl \rightarrow Li_3B_7O_{12} + 0.5\ O_2 + 0.5\ Cl_2 + 3\ Li$
3.881	−0.027	$Li_6B_7O_{13}Cl \rightarrow 3.5\ B_2O_3 + 1.25\ O_2 + 0.5\ Cl_2 + 6\ Li$
5.000	−0.347	$Li_6B_7O_{13}Cl \rightarrow 3.5\ B_2O_3 + 1.25\ O_2 + 0.5\ Cl_2 + 6\ Li$
(c) $Li_4Al_3B_4O_{12}Cl$		
ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−0.520	$Li_4Al_3B_4O_{12}Cl + 28.33\ Li \rightarrow 3\ Li_2Al + 12\ Li_2O + 1.333\ LiB_3 + LiCl$
0.079	−0.408	$Li_4Al_3B_4O_{12}Cl + 13.33\ Li \rightarrow 3\ Li_5AlO_4 + 1.333\ LiB_3 + LiCl$
0.353	−0.225	$Li_4Al_3B_4O_{12}Cl + 6.667\ Li \rightarrow 2\ Li_3BO_3 + 3\ LiAlO_2 + 0.667\ LiB_3 + LiCl$
0.661	−0.122	$Li_4Al_3B_4O_{12}Cl + 5.647\ Li \rightarrow 1.882\ Li_3BO_3 + 0.353\ B_6O + 3\ LiAlO_2 + LiCl$
0.860	−0.066	$Li_4Al_3B_4O_{12}Cl + 2.909\ Li \rightarrow 2.909\ LiBO_2 + 0.182\ B_6O + 3\ LiAlO_2 + LiCl$
1.284	−0.004	$Li_4Al_3B_4O_{12}Cl + 1.164\ Li \rightarrow 3.564\ LiBO_2 + 0.073\ B_6O + 0.6\ LiAl_5O_8 + LiCl$
1.358	0.000	$Li_4Al_3B_4O_{12}Cl \rightarrow Li_4Al_3B_4O_{12}Cl$
3.629	0.000	$Li_4Al_3B_4O_{12}Cl \rightarrow 0.571\ Li_4B_7O_{12}Cl + 0.429\ LiCl + 0.321\ O_2 + 1.5\ Al_2O_3 + 1.286\ Li$
3.687	−0.004	$Li_4Al_3B_4O_{12}Cl \rightarrow 0.571\ Li_4B_7O_{12}Cl + 0.321\ O_2 + 1.5\ Al_2O_3 + 0.214\ Cl_2 + 1.714\ Li$
3.821	−0.015	$Li_4Al_3B_4O_{12}Cl \rightarrow 0.667\ Al_4(B_2O_5)_3 + 0.75\ O_2 + 0.167\ Al_2O_3 + 0.5\ Cl_2 + 4\ Li$
5.000	−0.251	$Li_4Al_3B_4O_{12}Cl \rightarrow 0.667\ Al_4(B_2O_5)_3 + 0.75\ O_2 + 0.167\ Al_2O_3 + 0.5\ Cl_2 + 4\ Li$
(d) $Li_6Al_3B_4O_{13}Cl$		
ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−0.434	$Li_6Al_3B_4O_{13}Cl + 28.330\ Li \rightarrow 3\ Li_2Al + 13\ Li_2O + 1.333\ LiB_3 + LiCl$
0.079	−0.328	$Li_6Al_3B_4O_{13}Cl + 13.330\ Li \rightarrow 3\ Li_5AlO_4 + Li_2O + 1.333\ LiB_3 + LiCl$
0.311	−0.180	$Li_6Al_3B_4O_{13}Cl + 12.220\ Li \rightarrow 3\ Li_5AlO_4 + 0.333\ Li_3BO_3 + 1.222\ LiB_3 + LiCl$
0.353	−0.155	$Li_6Al_3B_4O_{13}Cl + 5.556\ Li \rightarrow 2.333\ Li_3BO_3 + 3\ LiAlO_2 + 0.556\ LiB_3 + LiCl$
0.661	−0.074	$Li_6Al_3B_4O_{13}Cl + 4.706\ Li \rightarrow 2.235\ Li_3BO_3 + 0.294\ B_6O + 3\ LiAlO_2 + LiCl$
0.860	−0.029	$Li_6Al_3B_4O_{13}Cl + 1.455\ Li \rightarrow 3.455\ LiBO_2 + 0.091\ B_6O + 3\ LiAlO_2 + LiCl$
1.284	0.000	$Li_6Al_3B_4O_{13}Cl \rightarrow Li_6Al_3B_4O_{13}Cl$
3.360	0.000	$Li_6Al_3B_4O_{13}Cl \rightarrow 4\ LiBO_2 + 0.600\ LiAl_5O_8 + LiCl + 0.100\ O_2 + 0.400\ Li$
3.507	−0.003	$Li_6Al_3B_4O_{13}Cl \rightarrow 4\ LiBO_2 + LiCl + 0.250\ O_2 + 1.5\ Al_2O_3 + Li$
3.687	−0.031	$Li_6Al3B_4O_{13}Cl \rightarrow 0.571\ Li_4B7O12Cl + 0.821\ O_2 + 1.500\ Al_2O_3 + 0.214\ Cl_2 + 3.714\ Li$
3.821	−0.054	$Li_6Al3B_4O_{13}Cl \rightarrow 0.667\ Al_4(B_2O_5$
5.000	−0.391	$Li_6Al3B_4O_{13}Cl \rightarrow 0.667\ Al_4(B_2O_5$
(e) $Li_4B_7S_{12}Cl$		
ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−1.821	$Li_4B_7S_{12}Cl + 23.33\ Li \rightarrow 2.333\ LiB_3 + 12\ Li_2S + LiCl$
0.866	−0.811	$Li_4B_7S_{12}Cl + 21\ Li \rightarrow 12\ Li_2S + LiCl + 7\ B$
1.595	−0.045	$Li_4B_7S_{12}Cl + 1.615\ Li \rightarrow 0.923\ Li_5B_7S_{13} + LiCl + 0.539\ B$
1.606	−0.044	$Li_4B_7S_{12}Cl + 1.167\ Li \rightarrow 0.833\ Li_5B_7S_{13} + 1.167\ BS + LiCl$
2.360	0.000	$Li_4B_7S_{12}Cl \rightarrow Li_4B_7S_{12}Cl$
2.491	0.000	$Li_4B_7S_{12}Cl \rightarrow 5\ BS_2 + 2\ BS + LiCl + 3\ Li$
2.686	−0.029	$Li_4B_7S_{12}Cl \rightarrow 5.333\ BS_2 + 1.333\ BS + 0.333\ BCl_3 + 4\ Li$
5.000	−0.492	$Li_4B_7S_{12}Cl \rightarrow 5.333\ BS_2 + 1.333\ BS + 0.333\ BCl_3 + 4\ Li$
(f) $Li_6B_7S_{13}Cl$		
ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−1.692	$Li_6B_7S_{13}Cl + 23.33\ Li \rightarrow 2.333\ LiB_3 + 13\ Li_2S + LiCl$
0.866	−0.730	$Li_6B_7S_{13}Cl + 21\ Li \rightarrow 13\ Li_2S + LiCl + 7\ B$
1.595	0.000	$Li_6B_7S_{13}Cl \rightarrow Li_6B_7S_{13}Cl$
2.454	0.000	$Li_6B_7S_{13}Cl \rightarrow 6\ BS_2 + BS + LiCl + 5\ Li$
2.686	−0.055	$Li_6B_7S_{13}Cl \rightarrow 6.333\ BS_2 + 0.333\ BS + 0.333\ BCl_3 + 6\ Li$
5.000	−0.716	$Li_6B_7S_{13}Cl \rightarrow 6.333\ BS_2 + 0.333\ BS + 0.333\ BCl_3 + 6\ Li$

TABLE IX. (*Continued.*)

(g) $Li_4Al_3B_4S_{12}Cl$		
ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−1.588	$Li_4Al_3B_4S_{12}Cl + 28.33\ Li \rightarrow 3\ Li_2Al + 1.333\ LiB_3 + 12\ Li_2S + LiCl$
0.126	−1.409	$Li_4Al_3B_4S_{12}Cl + 25.33\ Li \rightarrow 3\ LiAl + 1.333\ LiB_3 + 12\ Li_2S + LiCl$
0.351	−1.124	$Li_4Al_3B_4S_{12}Cl + 23.33\ Li \rightarrow\ LiAl_3 + 1.333\ LiB_3 + 12\ Li_2S + LiCl$
0.354	−1.121	$Li_4Al_3B_4S_{12}Cl + 22.33\ Li \rightarrow 1.333\ LiB_3 + 12\ Li_2S + LiCl + 3\ Al$
0.542	−0.911	$Li_4Al_3B_4S_{12}Cl + 21\ Li \rightarrow 2\ AlB_2 + 12\ Li_2S + LiCl + Al$
1.003	−0.427	$Li_4Al_3B_4S_{12}Cl + 18\ Li \rightarrow\ Li_5AlS_4 + 2\ AlB_2 + 8\ Li_2S + LiCl$
1.075	−0.362	$Li_4Al_3B_4S_{12}Cl + 12\ Li \rightarrow 3\ Li_5AlS_4 + LiCl + 4\ B$
1.664	−0.008	$Li_4Al_3B_4S_{12}Cl + 4\ Li \rightarrow 1.400\ Li_5AlS_4 + 4\ BS + 0.800\ Al_2S_3 + LiCl$
1.706	0.000	$Li_4Al_3B_4S_{12}Cl\ \rightarrow\ Li_4Al_3B_4S_{12}Cl$
2.122	0.000	$Li_4Al_3B_4S_{12}Cl\ \rightarrow 0.571\ Li_5B7S_{13} + 1.5\ Al_2S_3 + LiCl + 0.071\ S + 0.143\ Li$
2.415	−0.002	$Li_4Al_3B_4S_{12}Cl\ \rightarrow 0.5\ Li_5B7S_{13} + 0.5\ BS_2 + 1.5\ Al_2S_3 + LiCl + 0.5\ Li$
2.454	−0.003	$Li_4Al_3B_4S_{12}Cl\ \rightarrow 3.5\ BS_2 + 0.5\ BS + 1.5\ Al_2$
2.457	−0.004	$Li_4Al_3B_4S_{12}Cl\ \rightarrow 4\ BS_2 + 1.333\ Al_2S_3 + 0.333\ AlCl_3 + 4\ Li$
5.000	−0.512	$Li_4Al_3B_4S_{12}Cl\ \rightarrow 4\ BS_2 + 1.333\ Al_2S_3 + 0.333\ AlCl_3 + 4\ Li$
(h) $Li_6Al_3B_4S_{13}Cl$		
ϕ (V)	$\overline{\Delta\Omega}$ (eV/atom)	Reaction onset
0.000	−1.506	$Li_6Al_3B_4S_{13}Cl + 28.33\ Li \rightarrow 3\ Li_2Al + 1.333\ LiB_3 + 13\ Li_2S + LiCl$
0.126	−1.336	$Li_6Al_3B_4S_{13}Cl + 25.33\ Li \rightarrow 3\ LiAl + 1.333\ LiB_3 + 13\ Li_2S + LiCl$
0.351	−1.065	$Li_6Al_3B_4S_{13}Cl + 23.33\ Li \rightarrow\ LiAl_3 + 1.333\ LiB_3 + 13\ Li_2S + LiCl$
0.354	−1.061	$Li_6Al_3B_4S_{13}Cl + 22.33\ Li \rightarrow 1.333\ LiB_3 + 13\ Li_2S + LiCl + 3\ Al$
0.542	−0.862	$Li_6Al_3B_4S_{13}Cl + 21\ Li \rightarrow 2\ AlB_2 + 13\ Li_2S + LiCl + Al$
1.003	−0.400	$Li_6Al_3B_4S_{13}Cl + 18\ Li \rightarrow\ Li_5AlS_4 + 2\ AlB_2 + 9\ Li_2S + LiCl$
1.075	−0.339	$Li_6Al_3B_4S_{13}Cl + 12\ Li \rightarrow 3\ Li_5AlS_4 + Li_2S + LiCl + 4\ B$
1.595	−0.041	$Li_6Al_3B_4S_{13}Cl + 10.38\ Li \rightarrow 0.077\ Li_5B7S_{13} + 3\ Li_5AlS_4 + LiCl + 3.462\ B$
1.606	−0.036	$Li_6Al_3B_4S_{13}Cl + 10\ Li \rightarrow 3\ Li_5AlS_4 + BS + LiCl + 3\ B$
1.664	−0.008	$Li_6Al_3B_4S_{13}Cl + 4\ Li \rightarrow 1.8\ Li_5AlS_4 + 4\ BS + 0.6\ Al_2S_3 + LiCl$
1.706	0.000	$Li_6Al_3B_4S_{13}Cl\ \rightarrow\ Li_6Al_3B_4S_{13}Cl$
2.117	0.000	$Li_6Al_3B_4S_{13}Cl\ \rightarrow 0.571\ Li_5B7S_{13} + 1.5\ Al_2S_3 + LiCl + 1.071\ S + 2.143\ Li$
2.415	−0.031	$Li_6Al_3B_4S_{13}Cl\ \rightarrow 4\ BS_2 + 1.5\ Al_2S_3 + LiCl + 0.5\ S + 5\ Li$
2.555	−0.064	$Li_6Al_3B_4S_{13}Cl\ \rightarrow 4\ BS_2 + 1.333\ Al_2S_3 + 0.333\ AlCl_3 + S + 6\ Li$
5.000	−0.762	$Li_6Al_3B_4S_{13}Cl\ \rightarrow 4\ BS_2 + 1.333\ Al_2S_3 + 0.333\ AlCl_3 + S + 6\ Li$

TABLE X. Equilibrium phases and changes in the grand canonical potential $\Delta\Omega$ for Li (thio)boracite electrolytes in an idealized battery configuration. The anode μ_{Li} is set to the chemical potential of bulk lithium, μ^0_{Li}, while the cathode μ_{Li} is adjusted to $\mu^0_{Li} - 5$ eV to simulate a high voltage of 5 V. The $\overline{\Delta\Omega}$ energies are reported in units of eV/atom.

Materials	Phases at $\phi = 0$ V	$\overline{\Delta\Omega}(\phi = 0\ V)$	Phases atr $\phi = 5$ V	$\overline{\Delta\Omega}(\phi = 5\ V)$
$Li_4B_7O_{12}Cl$	Li_2O, LiB_3, $LiCl$	−0.714	B_2O_3, O_2, Cl_2	−0.225
$Li_6B_7O_{13}Cl$	Li_2O, LiB_3, $LiCl$	−0.600	B_2O_3, O_2, Cl_2	−0.347
$Li_4Al_3B_4O_{12}Cl$	Li_2Al, Li_2O, LiB_3, $LiCl$	−0.519	$Al_4B_6O_{15}$, O_2, Al_2O_3, Cl_2	−0.251
$Li_6Al_3B_4O_{13}Cl$	Li_2Al, Li_2O, LiB_3, $LiCl$	−0.434	$Al_4B_6O_{15}$, O_2, Al_2O_3, Cl_2	−0.391
$Li_4B_7S_{12}Cl$	LiB_3, Li_2S, $LiCl$	−1.821	BS_2, BS, BCl_3	−0.492
$Li_6B_7S_{13}Cl$	LiB_3, Li_2S, $LiCl$	−1.692	BS_2, BS, BCl_3	−0.716
$Li_4Al_3B_4S_{12}Cl$	Li_2Al, LiB_3, Li_2S, $LiCl$	−1.588	BS_2, Al_2S_3, $AlCl_3$	−0.512
$Li_6Al_3B_4S_{13}Cl$	Li_2Al, LiB_3, Li_2S, $LiCl$	−1.506	BS_2, Al_2S_3, $AlCl_3$, S	−0.762

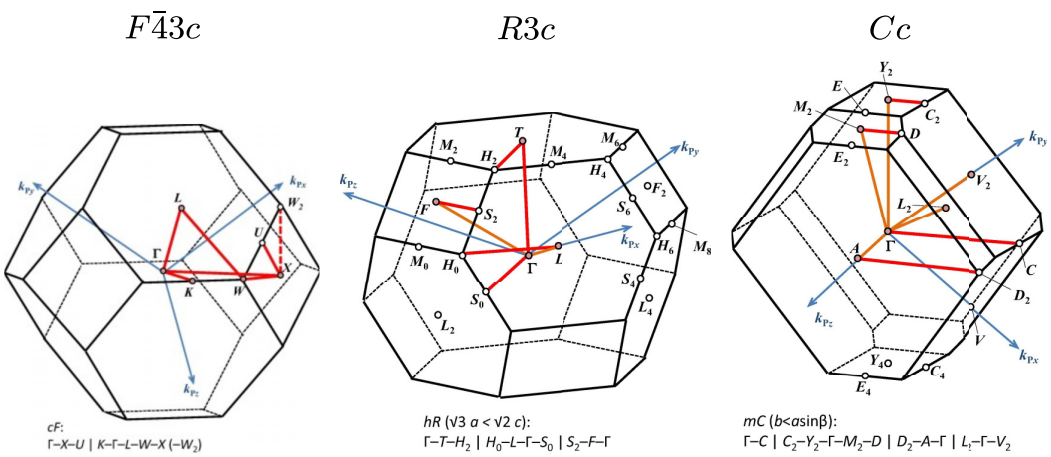


FIG. 19. Brillouin zone diagrams from Ref. [60] reproduced with permission from the publisher.

[1] W. Jeitschko, T. A. Bither, and P. E. Bierstedt, Acta Crystallogr. Sect. B **33**, 2767 (1977), detailed crystal parameters associated with this paper are available as cif files from various crystallography databases.

[2] D. Tan, F. Wang, T. Pietsch, M. A. Grasser, T. Doert, and M. Ruck, ACS Appl. Energy Mater. **2**, 5140 (2019).

[3] Y. Li and N. A. W. Holzwarth, Phys. Rev. Mater. **6**, 025401 (2022).

[4] K. Kajihara, N. Tezuka, M. Shoji, J. Wakasugi, H. Munakata, and K. Kanamura, Bull. Chem. Soc. Jpn. **90**, 1279 (2017).

[5] T. Katsumata, Y. Aoki, K. Fushimi, K. Otsuka, K. Ueda, and Y. Inaguma, Solid State Ionics **380**, 115921 (2022).

[6] K. Kaup, K. Bishop, A. Assoud, J. Liu, and L. F. Nazar, J. Am. Chem. Soc. **143**, 6952 (2021).

[7] Information Systems and Wake Forest University, WFU High Performance Computing Facility (2021).

[8] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[9] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[10] P. Giannozzi, S. de Gironcoli, P. Pavone, and S. Baroni, Phys. Rev. B **43**, 7231 (1991).

[11] X. Gonze, Phys. Rev. B **55**, 10337 (1997).

[12] X. Gonze and C. Lee, Phys. Rev. B **55**, 10355 (1997).

[13] S. Baroni, S. De Gironcoli, A. Dal Corso, and P. Giannozzi, Rev. Mod. Phys. **73**, 515 (2001).

[14] P. Giannozzi and S. Baroni, in *Handbook of Materials Modeling*, edited by S. Yip (Springer, New York, 2005), pp. 195–214.

[15] P. E. Blöchl, Phys. Rev. B **50**, 17953 (1994).

[16] J. P. Perdew, A. Ruzsinszky, G. I. Csonka, O. A. Vydrov, G. E. Scuseria, L. A. Constantin, X. Zhou, and K. Burke, Phys. Rev. Lett. **100**, 136406 (2008).

[17] N. A. W. Holzwarth, A. R. Tackett, and G. E. Matthews, Comput. Phys. Commun. **135**, 329 (2001).

[18] X. Gonze, F. Jollet, F. A. Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J.-M. Beuken, J. Bieder, A. Bokhanchuk *et al.*, Comput. Phys. Commun. **205**, 106 (2016).

[19] X. Gonze, B. Amadon, G. Antonius, F. Arnardi, L. Baguet, J.-M. Beuken, J. Bieder, F. Bottin, J. Bouchet, E. Bousquet *et al.*, Comput. Phys. Commun. **248**, 107042 (2020).

[20] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni *et al.*, J. Phys.: Condens. Matter **29**, 465901 (2017).

[21] P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, S. de Gironcoli, P. Delugas, F. Ferrari Ruffino *et al.*, J. Chem. Phys. **152**, 154105 (2020).

[22] Y. Li, Z. D. Hood, and N. A. W. Holzwarth, Phys. Rev. Mater. **4**, 045406 (2020).

[23] M. Born and K. Huang, *Dynamical Theory of Crystal Lattices* (Oxford University Press, Oxford, 1954); Appendix VIII expands the ideas presented by M. Born and R. Oppenheimer, Ann. d. Phys. **54**, 457 (1927).

[24] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, APL Mater. **1**, 011002 (2013).

[25] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton, JOM **65**, 1501 (2013).

[26] S. Kirklin, J. E. Saal, B. Meredig, A. Thompson, J. W. Doak, M. Aykol, S. Rühl, and C. Wolverton, npj Comput. Mater. **1**, 15010 (2015).

[27] H. T. Stokes and D. M. Hatch, J. Appl. Cryst. **38**, 237 (2005).

[28] K. Momma and F. Izumi, Appl Crystallogr. **44**, 1272 (2011), Code available from the website http://jp-minerals.org/vesta/en/.

[29] XMGRACE software available at the website, https://plasma-gate.weizmann.ac.il/pub/grace (2008).

[30] M. I. Aroyo, J. M. Perez-Mato, D. Orobengoa, E. Tasci, G. De La Flor, and A. Kirov, Bulg. Chem. Commun. **43**, 183 (2011), Also note that this space group information is also available from the International Union of Crystallography, which hosts the website for the International Tables for Crystallography, https://doi.org/10.1107/97809553602060000001.

[31] A. Levasseur, C. Fouassier, and P. Hagenmuller, Mater. Res. Bull. **6**, 15 (1971).

[32] A. Levasseur and D. J. Lloyd, J. Solid State Chem. **8**, 318 (1973).

[33] J.-M. Réau, A. Levasseur, G. Magniez, and B. Calès, Mater. Res. Bull. **11**, 1087 (1976).

[34] B. Calès, A. Levasseur, C. Fouassier, J. M. Réau, and P. Hagenmuller, Solid State Commun. **24**, 323 (1977).

[35] W. Jeitschko and T. A. Bither, Zeitschrift für Naturforschung B **27**, 1423 (1972).

[36] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevMaterials.8.065401 for Part I (SI1.pdf) includes detailed structural information, electronic partial density of states, and phonon band dispersions and density of states of the Li (thio)boracite materials. Additional Ref. [66] is cited. Part II (SI2.pdf) includes detailed structural information about ground-state structures of the related materials listed in Appendix B and Table VIII of the paper.

[37] S. P. Ong, L. Wang, B. Kang, and G. Ceder, Chem. Mater. **20**, 1798 (2008).

[38] M. Hellenbrandt, Crystallogra. Rev. **10**, 17 (2004).

[39] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, Comput. Mater. Sci. **68**, 314 (2013).

[40] W. Sun, S. T. Dacek, S. P. Ong, G. Hautier, A. Jain, W. D. Richards, A. C. Gamst, K. A. Persson, and G. Ceder, Sci. Adv. **2**, e1600225 (2016).

[41] M. Vlasse, A. Levasseur, and P. Hagenmuller, Solid State Ionics **2**, 33 (1981).

[42] N. Tezuka, Y. Okawa, K. Kajihara, and K. Kanamura, J. Ceram. Soc. Jpn. **125**, 348 (2017).

[43] Y. Zhu, X. He, and Y. Mo, ACS Appl. Mater. Interfaces **7**, 23685 (2015).

[44] W. D. Richards, L. J. Miara, Y. Wang, J. C. Kim, and G. Ceder, Chem. Mater. **28**, 266 (2016).

[45] Y. Zhu, X. He, and Y. Mo, J. Mater. Chem. A **4**, 3253 (2016).

[46] K. T. Butler, G. Sai Gautam, and P. Canepa, npj Comput. Mater. **5**, 19 (2019).

[47] T. Chen, G. Ceder, G. Sai Gautam, and P. Canepa, Front. Chem. **7**, 24 (2019).

[48] Y. Xiao, Y. Wang, S.-H. Bo, J. C. Kim, L. J. Miara, and G. Ceder, Nat. Rev. Mater. **5**, 105 (2020).

[49] M. K. Aydinol, A. F. Kohan, G. Ceder, K. Cho, and J. Joannopoulos, Phys. Rev. B **56**, 1354 (1997).

[50] S. P. Ong, O. Andreussi, Y. Wu, N. Marzari, and G. Ceder, Chem. Mater. **23**, 2979 (2011).

[51] M. V. Reddy, C. M. Julien, A. Mauger, and K. Zaghib, Nanomaterials **10**, 1606 (2020).

[52] Y. Okuno, J. Haruyama, and Y. Tateyama, ACS Appl. Energy Mater. **3**, 11061 (2020).

[53] Y. Tian, T. Shi, W. D. Richards, J. Li, J. C. Kim, S.-H. Bo, and G. Ceder, Energy Environ. Sci. **10**, 1150 (2017).

[54] D.-H. Wang, H.-Y. Zhou, C.-H. Hu, Y. Zhong, A. R. Oganov, and G.-H. Rao, Phys. Chem. Chem. Phys. **19**, 8471 (2017).

[55] A. A. Maradudin, E. W. Montroll, G. H. Weiss, and I. P. Ipatova, *Theory of Lattice Dynamics in the Harmonic Approximation*, 2nd ed., Solid state physics: Supplement (Academic Press, New York, 1971), Vol. 3.

[56] K. Parlinski, Z. Q. Li, and Y. Kawazoe, Phys. Rev. Lett. **78**, 4063 (1997).

[57] A. Togo, L. Chaput, T. Tadano, and I. Tanaka, J. Phys.: Condens. Matter **35**, 353001 (2023).

[58] A. Togo, J. Phys. Soc. Jpn. **92**, 012001 (2023).

[59] Y. Li, W. C. Kerr, and N. A. W. Holzwarth, J. Phys.: Condens. Matter **32**, 055402 (2020).

[60] Y. Hinuma, G. Pizzi, Y. Kumagai, F. Oba, and I. Tanaka, Comput. Mater. Sci. **128**, 140 (2017), online tool available from the website, https://www.materialscloud.org/work/tools/seekpath.

[61] In order to check the numerical accuracy of the imaginary modes in *R*3*c* $Li_6B_7S_{13}Cl$, we repeated the calculations checking the convergence parameters and also using density functional perturbation theory methodology with both ABINIT and QUANTUM ESPRESSO. Essentially the same results were obtained in all of the tests.

[62] G. Petretto, S. Dwaraknath, H. P.C. Miranda, D. Winston, M. Giantomassi, M. J. van Setten, X. Gonze, K. A. Persson, G. Hautier, and G.-M. Rignanese, Scientific Data **5**, 180065 (2018).

[63] K. Kaup, A. Assoud, J. Liu, and L. F. Nazar, Angew. Chem. Int. Ed. **60**, 6975 (2021).

[64] Y. Li and N. A. W. Holzwarth, Phys. Rev. Mater. **6**, 045403 (2022).

[65] W. Setyawan and S. Curtarolo, Comput. Mater. Sci. **49**, 299 (2010).

[66] L. E. Rush, Z. D. Hood, and N. A. W. Holzwarth, Phys. Rev. Mater. **1**, 075405 (2017).

Appendix C: Preprint: A study of the use of machine learning methodology for computing ionic conductivity in an example solid-state electrolyte material

A study of the use of machine learning methodology for computing ionic conductivity in an example solid state electrolyte material

D. Cory Lynch,[1] Chuin Wei Tan,[2] Boris Kozinsky,[2] and N. A. W. Holzwarth[1]

[1] *Department of Physics, Wake Forest University, Winston-Salem, NC 27109, USA*
[2] *John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA*

(Dated: 2026-08-03, 15:53:05)

Realistically simulating macroscopic phenomena from the knowledge of atomic scale interactions has challenged generations of computational scientists. In the present paper, we examine how machine learning methods can be used towards accomplishing the "ensemble sampling" envisioned for evaluating the Green-Kubo formulation of the full ionic conductivity of a solid electrolyte system. This work explores some of the advantages and challenges of using Allegro machine learned interatomic potentials together with traditional computational algorithms toward the goal of finding ways in which it can best be used to enhance computational methodogies. Enhanced computational methodolgies can lead to advances in understanding of basic phenomena through more realistic simulations of materials and their properties. In particular, the computational efficiency and accuracy of the Allegro trained interatomic potentials opens the possibility of performing simulations that can more closely represent conditions of experimental measurements..

I. INTRODUCTION

Recently, there have been a number of advances in methodologies for simulating ionic conductivity in materials from equilibrium molecular dynamics simulations and Green-Kubo linear response [1–4]. In addition to improvements in the theoretical formalism [5–8], several researchers have shown that augmenting the first principles investigations with machine learning interatomic potentials can be beneficial [9–11]. For example, Gigli and co-workers [9] used machine learning methods to perform nanosecond-long molecular dynamics simulations on crystalline Li_3PS_4 in order to study phase transformations and new mechanisms of ionic conduction. In another example, Harper and co-workers [10] showed that by using machine learning methods, it is possible to perform microsecond-long simulations of Li ion migration at physically realistic temperatures for crystalline and amorphous Li_3PS_4, providing valuable insight into the experimental ^{7}Li NMR measurements. Another example is provided by Minami and co-workers [11] who used machine learning techniques to, among other things, examine the full ionic conductivity of the Li superionic conductor $Li_{10}GeP_2S_{12}$ [12] using the Green-Kubo approach, also showing that long molecular dynamics simulation times at relatively low temperatures can yield numerically controlled results, consistent with non-equilibrium molecular dynamics methods that they are also developing. In general, while computational estimates of ionic conductivity based on the assumption of uncorrelated ionic motion have been quite successful such as described in Ref. [13], computing the full ionic conductivity from first principles approaches remains a challenge. This challenge is partly due to the computational demands of first principles formalisms and partly due to the complexity of the molecular dynamics simulations themselves. Here, we examine the extent to which machine-learned interatomic potentials can retain first-principles accuracy while offering the computational efficiency required to make Green–Kubo ensemble averaging computationally tractable.

In the present paper, we report the results of a systematic study of the utility and intricacies of using the Allegro [14, 15] interatomic interaction potential as a machine learned surrogate model trained on first-principles calculations to describe specific ionic conducting systems. These Allegro interaction potential models are then used within the "Large-scale Atomic/Molecular Massively Parallel Simulator" (LAMMPS) package [16] to evaluate the interaction potentials in terms of their structural predictions and to produce ion trajectories for evaluating the Green-Kubo equations for these systems. The focus of this work is to examine how the improved accuracy and efficiency of machine learning methods can produce molecular dynamics results for evaluating the Green-Kubo equations to estimate the full ionic conductivity.

All calculations were performed using the Wake Forest University High Performance Computing Facility [17] . In addition to using our in-house programs and scripts, we used several public domain software packages including QUANTUM ESPRESSO

[18, 19], NequIP [20], Allegro [14, 15], LAMMPS [16], and "Atomic Simulation Environment " ASE [21].

The remainder of the paper is organized as follows. Sec. II introduces the two test ionic material structures S1 and S2 used in the study. Sec. III outlines the basic equations used in simulating the ionic conductivity based on the Green-Kubo formalism and various methodologies for estimating the ensemble averages of molecular dynamics simulation results consistent with the Green-Kubo formulation. Sec. IV introduces some of the relevant machine learning concepts and details how the Allegro package [14, 15] is implemented in the present work. Sec. V discusses three important aspects of molecular dynamics simulations as they are used in the present work. Results and analyses are given in Sec. VI while concluding remarks are made in Sec. VII.

II. SOLID ELECTROLYTE TEST SYSTEMS

While the focus of the present work is on evaluating various aspects of the methodology, it is important to choose real and interesting ionic conducting materials for testing.

Solid state electrolytes formed from alloys of Li_3PO_4 and Li_4SiO_4 date back to the early work of Huggins and collaborators in the 1970's [22]. More recently, Deng and collaborators [23] found the most conductive stoichiometry of the alloy to be $(Li_3PO_4)_{3/4}(\ Li_4SiO_4)_{1/4}$ for which they reported a unit cell structure, having fractional occupancy sites, based on their experimental and modeling analyses. In the present work, we loosely based our two model systems, "S1" and "S2" from this unit cell. For structure S1, we took the Deng unit cell but chose only some of the determined sites with full occupancy and then constructed a $1 \times 2 \times 2$ supercell. For structure S2, some P and Si positions were switched. From these two 132 ion initial geometries, the reference equilibrium geometries for the S1 and S2 models, were found by optimizing each using the QUANTUM ESPRESSO software package. [18, 19] These calculations used the PBESOL exchange-correlation functional [24] and the Projector Augmented Wave (PAW) formalism of Blöchl [25] with PAW atomic datasets generated by ATOMPAW [26]. Example input tolerance and convergence settings for the QUANTUM ESPRESSO runs are given in the Supplemental Materials [27]. The QUANTUM ESPRESSO optimization calculations found that for the two 132 atom unit cells, reference structure S2 has an equilibrium energy 0.31 eV higher than that of S1 and an equilibrium volume of 2.94 $Å^3$ larger than that of S1. The structural parameters of the two equilibrium reference structures are also listed in the Supplemental Materials [27] and the structures are visualized in Fig. 1.

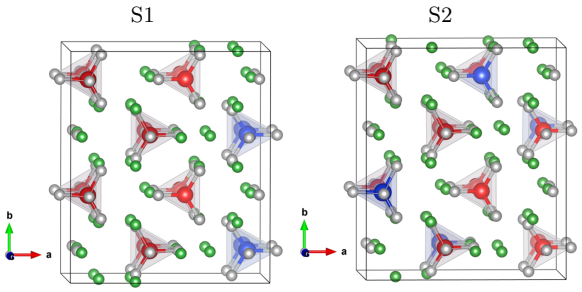


FIG. 1. Ball and stick visualization of the reference structures S1 and S2. Ball colors are green, blue, red, and grey representing Li, Si, P, and O, respectively. The polyhedron shading is used to indicate the PO_4 and SiO_4 tetrahedra.

III. THEORETICAL FORMULATIONS AND COMPUTATIONAL STRATEGIES FOR IONIC CONDUCTIVITY

The equations presented in this section summarize developments reported in the literature concerning the general formalism for calculating the ionic conductivity of an ionic material based on a classical mechanics treatment, from a knowledge of the charges, positions, and velocities of the particles as a function of time. In principle, the equations can be evaluated from a knowledge of the interparticle interaction potential Φ whether that knowledge comes from empirical potential models, density functional theory [28, 29], or as in the present work, from machine-learned models in various ways.

A. The Green-Kubo equation and various approximations for evaluating ionic conductivity

Within the context of first-principles methodologies and thanks to the Born-Oppenheimer approximation [30, 31], it is reasonable to treat the ionic conductivity in terms of classical mechanics. For electronically insulating and charge neutral materials, Grasselli and collaborators [6, 7] show that the ionic conductivity should be calculated from the classical ionic current density contributed by all of

the N_I ions i within the unit simulation cell of the system having volume Ω:

$$\mathbf{J}(t) = \frac{1}{\Omega}\sum_{i=1}^{N_I} q_i \mathbf{v}_i(t) \;\text{ with }\; \sum_{i=1}^{N_I} q_i = 0. \tag{1}$$

Here q_i denotes the ionic charge of species i and $\mathbf{v}_i(t)$ denotes its velocity at time t. Typically, as recently shown by Grasselli and Baroni [6] from the more rigorous formulation in terms of Born effective charges, q_i is well represented by the integer oxidation state of the ion i multiplied by the elementary charge e. Because of charge neutrality, the value of the current is independent of the possibly time-dependent reference frame used in simulations.

Grasselli and Baroni [7], as well as many textbooks [32–34] note that there often are numerical advantages to modifying the evaluation in terms of the time integral of the current density

$$\int_0^t \mathbf{J}(t')dt' = \mathbf{P}(t) - \mathbf{P}(0), \tag{2}$$

where $\mathbf{P}(t)$ denotes a dipole density as a function of t. The dipole density $\mathbf{P}(t)$ can be directly expressed in terms of a summation involving the positions of each ion $\mathbf{r}_i(t)$ as a function of t:

$$\mathbf{P}(t) = \frac{1}{\Omega}\sum_{i=1}^{N_I} q_i \mathbf{r}_i(t). \tag{3}$$

Again, because of the charge neutrality of the system, the full dipole density $\mathbf{P}(t)$ is well-defined in terms of possibly time dependent reference frame of the computation for the same reasons that that the full current density $\mathbf{J}(t)$ is also well-defined.

The relationship of atomic level interactions to macroscopic measurements, spanning 10^{23} orders of magnitude, is not trivial. Aided by the 1930's work of Lars Onsager [35, 36] on the "Reciprocal Relations in Irreversible Processes", the Green-Kubo [1–4] formula for estimating the ionic conductivity from long simulations of atomic properties was introduced, averaging over many possible ensembles at equilibrium. In terms of the polarization density, the Green-Kubo formula for the directionally averaged full ionic conductivity σ^{GK} is determined from

$$\sigma^{\mathrm{GK}} T \equiv$$
$$\sigma^{\mathrm{full}} T = \frac{\Omega}{6k_B}\lim_{\tau\to\infty}\frac{\left\langle |\mathbf{P}(\tau) - \mathbf{P}(0)|^2 \right\rangle_{\mathrm{ensembles}}}{\tau}. \tag{4}$$

Here, for convenience in our analysis, we have written the expression in terms of the conductivity $\times$ temperature product with k_B denoting the Boltzmann constant, T denoting the thermodynamic temperature of the system, and τ representing a time interval between successive evaluations of $\mathbf{P}(t)$. The angular brackets mean that the expression is to averaged over many ensembles of the the system under consideration. After performing the ensemble average for each time interval τ, the expression must be valuated in the infinite τ limit.

For many systems of interest, not all of the ions of the system contribute directly to the conductivity. For example, in the simulation cell, there may be one diffusing species of ion D (Li^+ in our example). Denoting the charge of each D ion as q_D, we have

$$\sum_{i\in D} q_i = N_D q_D \equiv q_D^{\mathrm{total}}, \tag{5}$$

where N_D denotes the number of diffusing ions in the simulation cell. The remaining ions (Si^{4+}, P^{5+}, and O^{2-} in our example) are part of a non-diffusing framework composed of ions having various charges q_j and total charge q_F^{total}. The framework charges are given by

$$\sum_{j\in F} q_j = q_F^{\mathrm{total}} = -q_D^{\mathrm{total}}, \tag{6}$$

where the last equality is a result of charge neutrality of the system. It is convenient to re-express the dipole density given in Eq. (3) in terms of the diffusing ions alone, using an appropriate reference position, as discussed by Marcolongo *et al.* [5]. In this case, a well defined reference is $\mathbf{r}_{\mathrm{corr}}(t)$, the center of charge of the framework at each time step t:

$$\mathbf{r}_{\mathrm{corr}}(t) = \frac{1}{q_F^{\mathrm{total}}}\sum_{j\in F} q_j \mathbf{r}_j(t) = -\frac{1}{N_D q_D}\sum_{j\in F} q_j \mathbf{r}_j(t). \tag{7}$$

The corresponding diffusing ion dipole density can be calculated from

$$\widetilde{\mathbf{P}}_D(t) = \frac{q_D}{\Omega}\sum_{i\in D}\left(\mathbf{r}_i(t) - \mathbf{r}_{\mathrm{corr}}(t)\right) \equiv \frac{q_D}{\Omega}\sum_{i\in D}\Delta\mathbf{r}_i(t). \tag{8}$$

It is apparent that $\widetilde{\mathbf{P}}_D(t)$ defined in Eq. (8) is numerically equal to the dipole density determined from Eq. (3) and can be used to calculate the full ionic conductivity within the Green-Kubo formalism

Eq. (4). The explicit expression is

$$\sigma^{\rm full}T = \frac{\Omega}{6k_B}\lim_{\tau\to\infty}\frac{\left\langle\left|\widetilde{\mathbf{P}}_D(\tau)-\widetilde{\mathbf{P}}_D(0)\right|^2\right\rangle_{\rm ensembles}}{\tau}$$
$$= \frac{q_D^2}{6k_B\Omega}\lim_{\tau\to\infty}\frac{\left\langle\left|\sum_{i\in D}\left(\Delta\mathbf{r}_i(\tau)-\Delta\mathbf{r}_i(0)\right)\right|^2\right\rangle_{\rm ensembles}}{\tau} \tag{9}$$

For diffusing ions whose motions are uncorrelated, the ionic conductivity can also be estimated from their mean squared displacement (MSD(τ)) which can be correspondingly calculated from

$$\mathrm{MSD}(\tau) = \left\langle\sum_{i\in D}\left|\Delta\mathbf{r}_i(\tau)-\Delta\mathbf{r}_i(0)\right|^2\right\rangle_{\rm ensembles}. \tag{10}$$

The corresponding conductivity due to uncorrelated diffusing ions – "tracer conductivity" ($\sigma^{\rm tr}$) can be estimated in terms of the conductivity times temperature product according to

$$\sigma^{\rm tr}T = \frac{q_D^2}{6k_B\Omega}\lim_{\tau\to\infty}\frac{\mathrm{MSD}(\tau)}{\tau}. \tag{11}$$

Here we see that the full and tracer conductivity expressions are similar except that contributions of the form

$$\left(\Delta\mathbf{r}_i(\tau)-\Delta\mathbf{r}_i(0)\right)\cdot\left(\Delta\mathbf{r}_j(\tau)-\Delta\mathbf{r}_j(0)\right) \quad \text{for } i\neq j \tag{12}$$

are *included* in Eq. (9) and *excluded* in Eq. (10). From these expressions, it is possible to calculate the temperature dependent Haven ratio [37–40]

$$H_R(T) = \frac{\sigma^{\rm tr}(T)}{\sigma^{\rm full}(T)}. \tag{13}$$

The Haven ratio is often interpreted as a measure of correlation among the diffusing ions for values $H_R(T)\neq 1$.

We note that the definition of MSD(τ) given in Eq. (10) includes the time dependent reference position $\mathbf{r}_{\rm corr}(t)$ in order to be consistent with a full conductivity calculation as discussed above. This logical consistency motivates our inclusion of $\mathbf{r}_{\rm corr}(t)$ in our simulations, although it is our experience that its variation throughout a simulation is often small compared with the motions of the diffusing ions.

B. Strategies for performing ensemble averaging

One can think of various levels of averaging that can be used to approximate the ensemble sampling needed to approach the intent of the Green-Kubo formula Eq. (4) and (9) and the analogous tracer conductivity Eq. (11). Here we discuss three different contributions. In Sec. III B 1 the analysis of a given molecular dynamics simulation run in terms of the multiple examples of the time intervals τ is discussed. In Sec. III B 2 the possibility of also averaging similar molecular dynamics runs having different initial conditions is discussed. In Sec. III B 3, we discuss the possibility of additionally averaging over multiple similar Allegro potential models. In Sec. III B 4 the analysis of conductivity in terms of its Arrhenius temperature dependence is discussed. In addition to the trajectory configurations $\{\mathbf{r}_i(t)\}$, depending on the details of the molecular dynamics algorithm used, other quantities may also vary with time such as temperature $T(t)$ and volume $\Omega(t)$. It is a reasonable approximation to consider time variation of these system parameters separately from the analysis of particle trajectories. Also note that in this section it is assumed that the trajectory configuration positions $\{\mathbf{r}_i(t)\}$ are well determined from molecular dynamics simulations. As described in Sec. V B, the mathematics of non-linear equations can lead to additional complications.

1. *Interval averaging methodologies for a single molecular dynamics run*

Processing methods for long molecular dynamics runs has been discussed in the literature such as for example in Refs. [8, 41, 42]. In this work, we adopt the block averaging method described, for example, by Grasselli [8]. Note that in the literature, algorithms for "block averaging" may differ from that described here as noted in Ref. [43].

The interval averaging that needs to be performed in Eqs. (9) and (10) averages over many instances of the time interval τ that occurs within a given molecular dynamics run for a range of values $\tau_{\rm min}\leq\tau\leq\tau_{\rm max}$. The block method is designed to ensure a fixed number (M) of instances for each time interval τ used in the analysis and the choice of the molecular run time parameters $t_{\rm start}$ and $t_{\rm finish}$ must adjusted accordingly. In general, the block method does not access all of the available time steps of the run. Specifically, we will use the following notation to describe time during a molecular dynamics run. The simulation time t for each run has the range

$$t_{\rm initial}\leq t\leq t_{\rm final}. \tag{14}$$

In order to enable the simulation to approach the intended conditions after $t_{\rm initial}$, the trajectory anal-

ysis is started at $t_{\rm start} > t_{\rm initial}$, and performed in the time range

$$t_{\rm start} \le t \le t_{\rm finish}. \tag{15}$$

(Note that in some cases $t_{\rm final} = t_{\rm finish}$.) The molecular dynamics integration time step δt depends on the system which in our case was taken to be $\delta t = 0.0005$ ps for the LAMMPS runs. It is typical to choose the trajectory analysis sampling time Δt to be larger, which in our case was taken to be $\Delta t = 20\delta t$. For example, in terms of the sampling time interval of Δt, the integer M can be chosen such that

$$M \le \frac{1}{2}\frac{(t_{\rm finish} - t_{\rm start})}{\Delta t}. \tag{16}$$

In this way, it is possible perform an average over each time interval τ for $\tau_{\rm min} \le \tau \le \tau_{\rm max}$ of a given molecular dynamics trajectory relative to the beginning times, t_m for $m = 1 \dots M$:

$$t_m \equiv t_{\rm start} + (m-1)\Delta t. \tag{17}$$

For estimating the full ionic conductivity from a single molecular dynamics run at a target temperature for a given reference structure, it is convenient to define a block averaged interval function $\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)$ as follows, with a particular interaction potential Φ and initial velocity distribution by v. This is expressed in terms of the diffusing dipole density given in Eq. (8), based on averaging over instances of the time interval τ is given below followed by the expression for the squared "standard error" or the squared "standard deviation of the mean" [44]. Note that the single overbar is used to indicate a single average of the function for each time interval τ.

$$\overline{\Lambda}^{\rm full}_{\Phi v}(\tau) \equiv \frac{\Omega}{6k_B\tau}\left\langle |\widetilde{\mathbf{P}}_D(\tau) - \widetilde{\mathbf{P}}_D(0)|^2 \right\rangle_{\rm intervals} = \frac{1}{M}\frac{\Omega}{6k_B\tau}\sum_{m=1}^{M}\left(|\widetilde{\mathbf{P}}_D(\tau+t_m) - \widetilde{\mathbf{P}}_D(t_m)|^2\right). \tag{18}$$

$$\left(\Delta\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)\right)^2 = \frac{1}{M(M-1)}\sum_{m=1}^{M}\left(\frac{\Omega}{6k_B\tau}|\widetilde{\mathbf{P}}_D(\tau+t_m) - \widetilde{\mathbf{P}}_D(t_m)|^2 - \overline{\Lambda}^{\rm full}_{\Phi v}(\tau)\right)^2. \tag{19}$$

We note that the extra factor of M in this form of the error measure tends to reduce the magnitude of the measure significantly, especially in situations such as some of our examples, in which $M \approx 10^5$. However, for consistency, we adapted this formula throughout.

Similarly, for estimating the tracer ionic conductivity from a single molecular dynamics run in terms of the MSD(τ) in Eq. (10), based on averaging over instances of the time interval τ, it is convenient to define a function $\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau)$

$$\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau) \equiv \frac{q_D^2}{6k_B\Omega\tau}\left\langle \sum_{i\in D}|\Delta\mathbf{r}_i(\tau) - \Delta\mathbf{r}_i(0)|^2 \right\rangle_{\rm intervals} = \frac{1}{M}\frac{q_D^2}{6k_B\Omega\tau}\sum_{m=1}^{M}\left(\sum_{i\in D}|\Delta\mathbf{r}_i(\tau+t_m) - \Delta\mathbf{r}_i(t_m)|^2\right). \tag{20}$$

The squared standard error of $\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau)$ can be computed using a formula analogous to Eq. (19).

Estimating the ensemble averaging needed to evaluate the Green-Kubo Eq. (9) or corresponding tracer conductivity corresponding to Eq. (11) by averaging over instances of each interval τ could be calculated according to

$$\overline{\sigma T}\rfloor^{\chi}_{\Phi v} = \lim_{\tau\to\infty}\overline{\Lambda}^{\chi}_{\Phi v}(\tau) \equiv \overline{\Lambda}^{\chi}_{\Phi v}(\infty), \tag{21}$$

where the superscript χ is given by "full" or "tr" as appropriate.

In order to evaluate the full and tracer ionic conductivities in Eq. (21), an appropriate temperature is needed. A reasonable approximation is the average temperature for the molecular dynamics run as determined from

$$\overline{T}_{\Phi v} \equiv \frac{1}{\mathcal{M}}\sum_{t=t_{\rm start}}^{t_{\rm finish}} T(t) \;\text{ where }\; \mathcal{M} \equiv \frac{t_{\rm finish} - t_{\rm start}}{\Delta t}. \tag{22}$$

The squared standard error of the temperature relative to its average during the molecular dyanmics

run can be determined from [44]

$$\left(\Delta\overline{T}_{\Phi v}\right)^2 \equiv \frac{1}{\mathcal{M}(\mathcal{M}-1)} \sum_{t=t_{\rm start}}^{t_{\rm finish}} \left(T(t) - \overline{T}_{\Phi v}\right)^2 . \tag{23}$$

Note that for the block averaging method as implemented in this work, the effective number of time steps $\mathcal{M}$ for evaluating the temperature is roughly twice as large as the number of instances M of each time interval τ.

With the help of machine learned interatomic potentials, plots of the singly averaged interval functions $\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)$ and $\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau)$ for a large range of intervals τ are given in Sec. VI C. Here it is seen that even for a very large number of instances of the sampling intervals (M $\approx 10^5$), while the interval averaging is helpful, it may not be sufficient to approximate the ensemble average in a numerically controlled manner, particularly for the full Green-Kubo ionic conductivies and particularly for higher simulation temperatures.

2. *Additional averaging over initial velocity distributions*

In order to improve upon the ensemble averaging provided by interval averaging alone, it is worth considering additional averaging over initial velocity distributions for each target temperature and reference structure S1 or S2. Denoting the distinct initial velocity distributions with index v for $v = 1, 2, \ldots v_T$, it is convenient to define doubly averaged interval functions for the full and tracer conductivities, given by

$$\overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau) \equiv \frac{1}{v_T} \sum_{v=1}^{v_T} \overline{\Lambda}^{\chi}_{\Phi v}(\tau). \tag{24}$$

The squared standard error of the doubly averaged interval function for the full or tracer conductivity can be estimated from [44]

$$\begin{aligned} &\left(\Delta\overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau)\right)^2 = \\ &\frac{1}{v_T(v_T-1)} \sum_{v=1}^{v_T} \left(\overline{\Lambda}^{\chi}_{\Phi v}(\tau) - \overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau)\right)^2 \\ &\quad + \frac{1}{v_T} \sum_{v=1}^{v_T} \left(\Delta\overline{\Lambda}^{\chi}_{\Phi v}(\tau)\right)^2 \end{aligned} \tag{25}$$

The last term of this expression represents a rough estimate of the accumulating squared standard error of the singly averaged $\overline{\Lambda}^{\chi}_{\Phi}(\tau)$ function (due to the block algorithm in Eq. (19)) and in our experience, is numerically much smaller than the first term, which corresponds to the squared standard error of double average itself.

The conductivity × temperature product for the full or tracer conductivity is given by

$$\overline{\overline{\sigma T}}\rceil^{\chi}_{\Phi} = \lim_{\tau\to\infty} \overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau) \equiv \overline{\overline{\Lambda}}^{\chi}_{\Phi}(\infty). \tag{26}$$

In order to estimate the full and tracer conductivities from the double averaged results in Eq. (26), the double averaged temperature can be estimated according to

$$\overline{\overline{T}}_{\Phi} \equiv \frac{1}{v_T} \sum_{v=1}^{v_T} \overline{T}_{\Phi v}, \tag{27}$$

with a squared standard error estimate of

$$\begin{aligned} \left(\Delta\overline{\overline{T}}_{\Phi}\right)^2 \equiv &\frac{1}{v_T(v_T-1)} \sum_{v=1}^{v_T} \left(\overline{T}_{\Phi v} - \overline{\overline{T}}_{\Phi}\right)^2 \\ &+ \frac{1}{v_T} \sum_{v=1}^{v_T} \left(\Delta\overline{T}_{\Phi v}\right)^2 . \end{aligned} \tag{28}$$

Plots of the double averaged interval functions are presented in Sec. VI C. These plots show that the double averaging procedure does produce results with more reasonable asymptotic behavior in the long interval τ limit.

3. *Additionally averaging over compatable Allegro potential models*

As shown in Sec. VI C, it is interesting to compare results derived from comparable Allegro potential models Φ, such as those based on the same training data and having the same hyperparameters, differing only in the random initialization of the learned parameters and in the specific configurations used for training and validation. For each target temperature and reference structure, variations in the doubly averaged interval functions $\overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau)$ are indicative of the range of ionic conductivity results across multiple similar Allegro models. In these cases, it may make sense to use these diverse results to further approximate the ensemble averaging of the Green-Kubo formula in Eq. (9) by defining triple averaged interval functions. The triple averaged interval functions for the full and tracer conductivities can be defined according to

$$\overline{\overline{\Lambda}}^{\chi}(\tau) \equiv \frac{1}{N_\Phi} \sum_{\Phi=1}^{N_\Phi} \overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau). \tag{29}$$

Here N_Φ denotes the number of comparable Allegro potential models. The squared standard error of these expressions can be evaluated by expressions similar to that of Eq. (25). Numerically, we find that the magnitude of the error here is dominated by the doubly averaged contributions, with the variation across Allegro models contributing much smaller errors.

Similarly, we can then define the triple averaged estimates of the conductivity × temperature products as

$$\overline{\overline{\overline{\sigma T}}}\rfloor^\chi = \lim_{\tau\to\infty} \overline{\overline{\Lambda}}^\chi(\tau) \equiv \overline{\overline{\Lambda}}^\chi(\infty). \tag{30}$$

These estimates similarly require a corresponding triply averaged temperature,

$$\overline{\overline{T}} \equiv \frac{1}{N_\Phi}\sum_{\Phi=1}^{N_\Phi} \overline{\overline{T}}_\Phi, \tag{31}$$

with a squared standard error estimate of

$$\begin{aligned}\left(\Delta\overline{\overline{T}}\right)^2 \equiv& \frac{1}{N_\Phi(N_\Phi-1)}\sum_{\Phi=1}^{N_\Phi}\left(\overline{\overline{T}}_\Phi - \overline{\overline{T}}\right)^2 \\ &+ \frac{1}{N_\Phi}\sum_{\Phi=1}^{N_\Phi}\left(\Delta\overline{\overline{T}}_\Phi\right)^2.\end{aligned} \tag{32}$$

4. *Estimating conductivity and activation energy from computed $\overline{\Lambda}$ and $\overline{\overline{\Lambda}}$ functions*

For a set of simulations at a specific target temperature for a given reference structure, we can numerically evaluate the asymptotic values of Eqs. (26) and (30) by fitting the $\overline{\Lambda}(\tau)$ and $\overline{\overline{\Lambda}}(\tau)$ functions to the simple functional form

$$f^\chi(\tau) \equiv f_0^\chi + f_1^\chi\left(\frac{1}{\tau}\right). \tag{33}$$

Here χ represents "tr" or "full" as appropriate. Then the corresponding tracer and full conductivities can be estimated according to

$$\overline{\overline{\sigma}}^\chi_\Phi \approx \lim_{\tau\to\infty}\frac{f^\chi(\tau)}{\overline{\overline{T}}_\Phi} \equiv \frac{f_0^\chi}{\overline{\overline{T}}_\Phi} \tag{34}$$

and

$$\overline{\overline{\sigma}}^\chi \approx \lim_{\tau\to\infty}\frac{f^\chi(\tau)}{\overline{\overline{T}}} \equiv \frac{f_0^\chi}{\overline{\overline{T}}}. \tag{35}$$

These conductivity estimates will have associated errors which include the error from the linear regression in addition to the errors from the double and triple averaging process. For example, the error in the doubly averaged conductivity estimate can be defined as:

$$\left(\Delta\overline{\overline{\sigma}}^\chi_\Phi\right)^2 \equiv \left(\frac{f_0^\chi}{\overline{\overline{T}}_\Phi}\right)^2\left[\left(\frac{\Delta f_0^\chi}{f_0^\chi}\right)^2 + \left(\frac{\Delta\overline{\overline{T}}_\Phi}{\overline{\overline{T}}_\Phi}\right)^2\right], \tag{36}$$

where Δf_0^χ is given by

$$\left(\Delta f_0^\chi\right)^2 \equiv \left(\delta f_0^\chi\right)^2 + \left(\Delta\overline{\overline{\Lambda}}^\chi_\Phi(\tau_{max})\right)^2. \tag{37}$$

Here we have introduced δf_0^χ, which is the error in the fit of the coefficient f_0^χ to the data. Note that to include the error from the data itself, we are taking the error at τ_{max}, as this should be a better estimate of the data error for large τ. Similar errors can be calculated for the triply averaged conductivity estimates. The error for the Haven ratio defined earlier in the section is calculated similarly to Eq. 36, which is based on error propagation techniques described in [44].

From the estimates of the full and tracer conductivities $\overline{\overline{\sigma}}^\chi_\Phi$ and $\overline{\overline{\sigma}}^\chi$, along with their corresponding temperatures $\overline{\overline{T}}_\Phi$ and $\overline{\overline{T}}$, it is possible estimate the activation energy E_a from the Arrhenius relationship [45], which expresses the temperature dependence of the conductivity × temperature product in the form

$$\sigma T = C_0\exp\left(-\frac{E_a}{k_B T}\right). \tag{38}$$

In practice E_a is determined by fitting the $\ln(\overline{\overline{\sigma T}}\rfloor^\chi_\Phi)$ or $\ln(\overline{\overline{\overline{\sigma T}}}\rfloor^\chi)$ data to the simple form

$$g^\chi(T) = g_0^\chi + g_1^\chi\frac{1000}{T}, \tag{39}$$

so that

$$E_a^\chi = -1000 k_B g_1^\chi. \tag{40}$$

When performing these fits, we are fitting to datapoints which are transformations of our temperature and conductivity estimates. For example, in the case of the doubly averaged conductivity estimates for a particular model Φ, we have datapoints of the form $\left(1000/\overline{\overline{T}}_\Phi, \ln(\overline{\overline{\sigma}}^\chi_\Phi\overline{\overline{T}}_\Phi)\right)$, both of which have some corresponding error that must be calculated by propagating the data error through the respective functional transformations according to [44]. We can define the error in the temperature transformed data

as

$$\left(\Delta\left[1000/\overline{\overline{T}}_\Phi\right]\right)^2 \equiv \left(\frac{1000}{(\overline{\overline{T}}_\Phi)^2}\right)^2 \left(\Delta\overline{\overline{T}}_\Phi\right)^2, \quad (41)$$

and in the conductivity transformed data as

$$\left(\Delta\left[\ln(\overline{\overline{\sigma}}^\chi_\Phi \overline{\overline{T}}_\Phi)\right]\right)^2 \equiv \left(\frac{1}{\overline{\overline{\sigma}}^\chi_\Phi \overline{\overline{T}}_\Phi}\right)^2 \left(\Delta\left[\overline{\overline{\sigma}}^\chi_\Phi \overline{\overline{T}}_\Phi\right]\right)^2, \quad (42)$$

where we have defined the squared error in the conductivity × temperature product as $\left(\Delta\left[\overline{\overline{\sigma}}^\chi_\Phi \overline{\overline{T}}_\Phi\right]\right)^2 \equiv (\Delta f_0^\chi)^2$, given in Eq. 37.

Given that we are estimating the activation energy by fitting the Arrhenius data to the linear form defined in Eq. 39, to estimate the error in the activation energy, we must include the error from the Arrhenius data being fit as well as the error in the fit coefficient, analogous to Eq.s 36 and 37. We define the error in the activation energy as

$$(\Delta E_a^\chi)^2 \equiv (1000 k_B)^2 (\Delta g_1^\chi)^2, \quad (43)$$

where Δg_1^χ is given by

$$(\Delta g_1^\chi)^2 \equiv (\delta g_1^\chi)^2 + \left(\Delta\left[\frac{\ln(\overline{\overline{\sigma}}^\chi_\Phi \overline{\overline{T}}_\Phi)}{1000/\overline{\overline{T}}_\Phi}\right]\right)^2. \quad (44)$$

The second term contains the contribution to the error in the slope from the data, and is calculated by propagating the error of the two transformed data values through the division operation, analogous to Eq. 36. Results for this analysis is presented in Sec. VI C.

IV. USING MACHINE LEARNING TO IMPROVE ESTIMATES OF IONIC CONDUCTIVITY

Machine learning methods are increasingly adopted in computational physics, including in studies of ionic conducting materials [9–11]. This trend is driven by the potential of machine-learned models to provide improved accuracy and computational efficiency in evaluating well-established quantities such as the Green-Kubo ionic conductivity.

In contrast to traditional computational techniques used to evaluate classical many-body interactions where the independent variables and their functional dependencies are well known and where powerful tools such as the variational principle produce unique results, machine learning methods require the expansion of computational methodologies. This expansion can be broadly described in two different categories – (1) the development of machine learning "architectures", which include designing smooth and differentiable functional forms to describe the atomic environments, and (2) the development of training methodologies for parsing large amounts of data in order to construct machine learned models consistent with physical laws (such as energy conservation) and maintaining data fidelity. Seminal machine learning interatomic potentials include Behler-Parrinello neural networks published in 2007 [46] and Gaussian approximation potentials [47] published in 2010, which established that atomic interaction energies could be learned reliably from quantum mechanical data. The field has since grown rapidly [48, 49]. Neural networks are particularly well-suited for learning potential energy surfaces needed to study the ionic materials of interest in the present work. The universal approximation theorems [50, 51] guarantee that sufficiently expressive networks can represent a wide class of continuous functions, including the complex many-body interactions present in condensed matter systems.

In terms of machine learning architectures, neural network models designed to describe interatomic interactions parametrize the system's configuration, comprising atomic positions $\{\mathbf{r}_i\}$ and species $\{Z_i\}$, with sufficiently flexible functional forms that are mapped to the potential energy of the system. A key conceptual advance was the recognition that these functional forms should be consistent with physical symmetries, including rotational, translational, and permutational invariance of atomic systems, and could be encoded directly into the architecture, giving rise to E(3)-equivariant neural networks. NequIP [20] was a pioneering architecture to adopt this approach, while Allegro [14] extended the approach with a strictly local formulation to enable efficient large-scale simulations [52]. Both are built on the `e3nn` library [53], which implements the equivariant operations underlying these architectures. Subsequent developments improved computational efficiency through compilation and custom GPU kernels [15, 54]. It is also important that the chosen neural network architecture and its contained descriptors produce model potentials Φ based on smooth and differentiable functions of the input atomic positions. This results in accurate model energies, forces, and stresses, avoiding discontinuities that may lead to unphysical simulation artifacts.

There have also been many advances in the devel-

opment of efficient algorithms for training machine learning models on large datasets. Some of these can be found in the PyTorch library [55] which is used in the Allegro software package. The training process involves minimizing a loss function for the training dataset, as described in Sec. IV D which also discusses various metrics that can be used to measure the training error. As a result of the large number of fitting values (learned parameters) and large number of atomic configurations to be represented, it is important to construct large training datasets which span the materials' configurations of interest. Random sampling is used due to the large dataset size and to reduce the probability of introducing bias in the fitting. Allegro divides the fully constructed input dataset into "training" and "validation" datasets [56] and [57]. While the model is expected to accurately represent training configurations, it is important that the model interpolate well to the untrained configurations. The validation dataset is used to estimate the ability of the model to interpolate reasonably well to untrained configurations and to "generalize" the model.

A. Choice of machine learning architecture - Allegro

In the present work, we adopted the Allegro machine learning interatomic potential model [14], a strictly local extension of the NequIP architecture [20]. NequIP is a message-passing graph neural network that passes messages between atoms through E(3)-equivariant convolution operations. NequIP propagates learned atom-centered representations that interact with their local environment comprising the learned representations of neighboring atoms within a fixed cutoff radius r_{cut}. This interaction involves iterative message-passing operations at each layer of the network, which extends the effective radius of the local environment of each atom. Raising the number of message-passing layers therefore increases the effective range of each atom's local potential surface to include effects from larger numbers of neighboring atoms.

While achieving impressive accuracies, the message-passing nature of NequIP poses difficulty in scaling up to large simulations across distributed computing resources [58]. The Allegro architecture addresses this limitation through a strictly local construction. Allegro propagates representations that describe atom-pairs, which interact with their local environment comprising other atom-pairs within a fixed cutoff radius r_{cut}. This interaction does not involve node-to-node message-passing, keeping the effective radius of the local environment fixed. Thus, the local potential surface for each atom-pair is strictly local, and only includes effects from pairs within r_{cut}. Both NequIP and Allegro utilize spherical tensors, an efficient way to represent atomic environments in an equivariant way, which are combined through iterated tensor products of representations, defining the interaction operation mentioned earlier.

In Allegro, the pair-centered representations are non-trivial learned constructions of radial and angular basis functions. The angular basis functions are the spherical harmonics $Y_{\ell m}$ which are functions of the direction of the pair, $\hat{\mathbf{r}}_{ik}$. Here, ℓ is an integer such that $0 \leq \ell \leq \ell_{max}$, and m is an integer such that $-\ell \leq m \leq \ell$. The radial basis functions R_b are functions of the distance between the atoms in the pair, r_{ik}, and are based on the work of Gasteiger and co-workers [59]. They take the form

$$R_b(r_{ik}) \equiv \left(\frac{\sin(\pi b r_{ik}/r_{cut})}{\pi b r_{ik}/r_{cut}} \right) u(r_{ik}/r_{cut}), \qquad (45)$$

where $u(r_{ik}/r_{cut})$ is a smooth polynomial cutoff function and b is an integer such that $1 \leq b \leq b_{max}$. The cutoff function $u(s)$ is takes the form

$$u(s) \equiv \begin{cases} 1 - 28s^6 + 48s^7 - 21s^8 & : \ 0 \leq s \leq 1 \\ 0 & : \ s > 0 \end{cases} \qquad (46)$$

Plots of the basis functions are shown in Fig. 2.

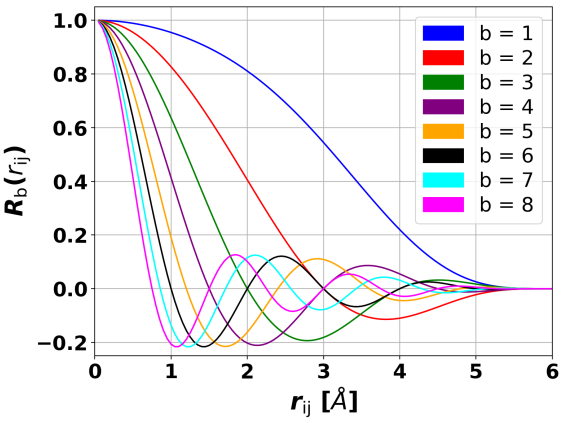


FIG. 2. Plot of the 8 radial basis functions $R_b(r_{ik})$ as defined in Eq. (45), scaled with $r_{\rm cut} = 6$ Å as used in this work.

These learned many-body representations of each atom-pair are then used to define the energy of the

pair, as seen in Ref. [14]. While a full treatment of the pair energy is beyond the scope of this work, we nonetheless provide a brief description in simpler notation here

$$E_{ij} = \Psi_{ij}(\{R_b(r_{ik})\}, \{Y_{\ell m}(\hat{\mathbf{r}}_{ik})\}). \quad (47)$$

Here we simply define the pair energy E_{ij} as some learned function Ψ_{ij}, a complicated function defined by the Allegro architecture. This function defines the potential surface for a particular pair of atoms indexed by ij and contains many learned parameters optimized during training. It is composed of the complex pair-centered representations containing the previously described radial and angular basis functions for each pair ik, where k indexes all atoms in the neighborhood of i within a defined cutoff radius r_{cut}, including j, such that $k \neq i$.

From the pair energies, we can then define the total system energy E_{system} as follows, as seen in Ref. [14].

$$E_{system} = \sum_{i=1}^{N} \left(\mu_{Z_i} + \sigma_{Z_i} \sum_{j \in \mathcal{N}(i)} \sigma_{Z_i Z_j} E_{ij} \right). \quad (48)$$

Here N denotes the number of atomic sites in the simulation cell and σ_{Z_i} and μ_{Z_i} denote species specific scale and shift parameters for a specific atom indexed by i. The energy for a site i is given by a sum of scaled pair energies E_{ij}, where j indexes all atoms in the neighborhood of i defined by the cutoff radius r_{cut} such that $|\mathbf{r}_i - \mathbf{r}_j| \leq r_{\text{cut}}$ and $j \neq i$. The scaling factor is given by $\sigma_{Z_i Z_j}$. The species specific scale and shift parameters, as well as the pair energy scaling factor, are optionally defined by the user.

The Allegro model is implemented in the Allegro software package, an "extension package" of the `nequip` software framework, which provides the underlying infrastructure for training and deploying machine learning interatomic potentials [15].

B. Settings used in Allegro training runs

The Allegro architecture is controlled by configuring its' hyperparameters, which in part defines the internal form of the pair potential surface Ψ_{ij} described in equation 47. Prominent hyperparameters of Allegro include the local neighborhood radial cutoff r_{cut}, the maximum rotational order for the spherical harmonics ℓ_{max}, the number of Allegro tensor product layers N_{layers}, the number of tensor features n_{tensor}, the number of scalar features n_{scalar}, and the dimensions of the various self contained feed-forward neural networks embedded within the architecture, $MLP_{dim} = d \times w$, where d is the depth of the network and w is the width. Moreover, there are also training hyperparameters that influence the learning dynamics and training process, such as the learning rate, stopping criteria, and optimization algorithm used.

TABLE I. Table showing parameter choices used for Allegro training runs.

Hyperparameter	Small model	Medium model
ℓ_{max}	1	2
n_{tensor}	16	32
n_{scalar}	32	64
$MLP_{dim}(d \times w)$	1×32	2×64
n_{params}	24500	91100

We have found that 4 main hyperparameters are most impactful on the measures of success of the Allegro model training and subsequent training and validation errors. These are r_{cut}, ℓ_{max}, n_{tensor} and MLP_{dim}. Error improvements are observed to saturate significantly past $r_{cut} = 6.0$ Å, therefore we have fixed $r_{cut} = 6.0$ Å. ℓ_{max} has non-linear computational cost increase for larger values, limiting the choice here – we typically used $\ell_{max} = 1, 2$. The MLP dimensions are less computationally intensive, and implicitly define the number of trainable parameters n_{params} in the final learned function. To construct realistic potential energy surfaces, the MLP must be flexible enough to capture the granularity of the surface, which requires the MLP dimensions be large enough, resulting in a fairly large number of learned parameters in the network. Although our hyperparameter choices may be suitable for this study, more generally the ideal hyperparameter choices are system dependent. We consider two model "sizes" we denote as "small" and "medium", which have the same hyperparameters and training settings besides a select few that differ which are in Table I.

We have found that the training settings which most directly impact model optimization are the initialization of the model prior to training and the early stopping settings, which determine training completes. Model initialization involves using one user-defined random seed to initialize the learned parameters throughout the network and another to split the input dataset into separate training and validation sets. The models are then trained until one of 3 early stopping criteria is met:

1. The maximum training time of 14 days is reached.

2. The maximum epoch of 14000 is reached.

3. The model fails to "improve" for 500 epochs.

Here the term epoch is defined as a single optimization step during the training process, and improvement in this context is defined as the model's validation set error reducing upon optimization. This is chosen in order that the "best" model, defined to be the one with the lowest validation set error, has the best interpolative capacity. In practice, the maxmum epoch is never met before 14 days, therefore only 1 and 3 affect the model's training completion. We typically see 2 types of stopping when training: a training "plateau" occurs, where the model stops improving and stops as a result before reaching the maximum training time, or the model simply runs out of time, which we call "timeout". The complete list of hyperparameters and training settings chosen is given in the Supplemental Materials [27].

C. Preparation of datasets for machine learning training

Density functional theory calculations, using the QUANTUM ESPRESSO code [18, 19] with the PBESOL exchange-correlation functional [24], were performed to generate quantum mechanical reference data for training and validation. The calculations were based on the Projector Augmented Wave (PAW) formalism of Blöchl [25] with PAW datasets generated by ATOMPAW [26]. Example input tolerance and convergence settings for the QUANTUM ESPRESSO runs are given in the Supplemental Materials [27].

In the course of our work on the project, we have developed some guidelines for creating effective input datasets for modeling ionic conductors.

1. The input dataset must have a sufficiently large set of representative configurations of the system. For our purposes, we determined this to be approximately 5000 configurations.

2. Since machine-learned models can only reliably predict properties represented in their training data, each training structure should include the atomic positions, total energy, atomic forces, and stress tensor for each configuration.

3. It is important to use stringent tolerance and convergence parameters for generating first-principles data such as those listed for the QUANTUM ESPRESSO input files given in the Supplemental Materials [27]. In practice, lower tolerance molecular dynamics runs computed with QUANTUM ESPRESSO were used to sample reasonable configurations, but were recomputed with the stringent tolerance and convergence parameters in order to obtain well converged forces and stresses before inclusion in the input dataset.

4. As mentioned in the Introduction, we were motivated to include low temperature configurations in our input dataset so that our trained models can accurately simulate low temperature molecular dynamics.

5. For reasons mentioned in Sec. V-B and in the recent literature [60], approximately 400 additional configurations representing relatively close pair distances were constructed for both structure S1 and S2 to help mitigate molecular dynamics instability.

6. In addition to the aforementioned approaches to improving coverage of the input configuration space, we also found it useful to include configurations from multiple starting structures for training a single model. For instance, training on combined configurations derived from both structures S1 and S2, in addition to training on configurations derived from structures S1 or S2 individually.

Based on these guidelines, we have constructed three input datasets for testing purposes, D_{in}^{S1+S2}, containing configurations derived from both structures S1 and S2, D_{in}^{S1}, containing configurations derived from only structure S1, and D_{in}^{S2}, containing configurations derived from only structure S2. These input datasets are split into training sets D_{train}^{S} and validation sets D_{val}^{S} internally with a ratio of $80:20$. Based on guideline 4, we have sampled from molecular dynamics runs computed with QUANTUM ESPRESSO at temperatures of $100K$, $400K$, $600K$, and $900K$ to create our input datasets for both structure S1 and S2. The higher temperature $600K$ and $900K$ runs were sampled at a ratio of $2:1$ relative to the $100K$ and $400K$ runs to ensure sufficient coverage of the input configuration space. For D_{in}^{S1+S2}, 2500 random configurations from each structure's sampled QUANTUM ESPRESSO molecular dynamics data were included, along with the

roughly 400 constructed close pair distance configurations for both structures, for a total of around 5800 configurations. For D_{in}^{S1} and D_{in}^{S2}, 5000 random configurations from that structure's sampled QUANTUM ESPRESSO molecular dynamics data were included, along with the roughly 400 constructed close pair distance configurations for that structure, for a total of about 5400 configurations.

D. Model optimization and dataset error metrics within Allegro

The training process involves minimizing the loss function $\mathcal{L}$ to find the learned parameters in the network which gives a good functional approximation to the training data. In general, the loss function of a dataset D (i.e., D = train or D = val) is given by

$$\mathcal{L}^D = \lambda_E E^D_{MSE} + \lambda_F F^D_{MSE} + \lambda_S S^D_{MSE} \tag{49}$$

This function is composed of "Mean Squared Error" (MSE) values for the dataset D of energy, force, and stress components E^D_{MSE}, F^D_{MSE}, and S^D_{MSE}. These are defined in equations 50, 51, and 52. It also contains $\{\lambda_E, \lambda_F, \lambda_S\}$, the set of normalized weights for the energy, force, and stress MSEs respectively, where $\lambda_E + \lambda_F + \lambda_S = 1$. We choose the loss components to be equally weighted, i.e. $\lambda_E = \lambda_F = \lambda_S = 1/3$.

$$E^D_{MSE} = \frac{1}{N_C}\sum_{\alpha=1}^{N_C} |\hat{E}_\alpha - E_\alpha|^2 \tag{50}$$

$$F^D_{MSE} = \frac{1}{3N_C N_A}\sum_{\alpha=1}^{N_C}\sum_{\beta=1}^{N_A}\sum_{\gamma=1}^{3} |\hat{F}_{\alpha,\beta,\gamma} - F_{\alpha,\beta,\gamma}|^2 \tag{51}$$

$$S^D_{MSE} = \frac{1}{9N_C}\sum_{\alpha=1}^{N_C}\sum_{\gamma=1}^{9} |\hat{S}_{\alpha,\gamma} - S_{\alpha,\gamma}|^2 \tag{52}$$

Here we have defined N_C to be the number of configurations in the dataset D, N_A to be the number of atoms in the system, $\hat{E}_\alpha$ to be the predicted system energy for configuration $\alpha \in D$, E_α to be the known energy for configuration $\alpha \in D$, $\hat{F}_{\alpha,\beta,\gamma}$ to be the predicted γ-th component of force acting on atom β in configuration $\alpha \in D$, $F_{\alpha,\beta,\gamma}$ to be the known γ-th component of force acting on atom β in configuration $\alpha \in D$, $\hat{S}_{\alpha,\gamma}$ to be the predicted γ-th component of stress acting on configuration $\alpha \in D$, $S_{\alpha,\gamma}$ to be the known γ-th component of stress acting on configuration $\alpha \in D$.

A convenient way to assess the quality of a model after optimization is to use the "Mean Absolute Error (MAE) values for a given training or validation dataset D. These include the energy MAE E^D_{MAE}, the force MAE F^D_{MAE}, the stress MAE S^D_{MAE}, and the weighted sum WS^D. These definitions are given in equations 53, 54, 55, and 56.

$$E^D_{MAE} = \frac{1}{N_C}\sum_{\alpha=1}^{N_C} |\hat{E}_\alpha - E_\alpha| \tag{53}$$

$$F^D_{MAE} = \frac{1}{3N_C N_A}\sum_{\alpha=1}^{N_C}\sum_{\beta=1}^{N_A}\sum_{\gamma=1}^{3} |\hat{F}_{\alpha,\beta,\gamma} - F_{\alpha,\beta,\gamma}| \tag{54}$$

$$S^D_{MAE} = \frac{1}{9N_C}\sum_{\alpha=1}^{N_C}\sum_{\gamma=1}^{9} |\hat{S}_{\alpha,\gamma} - S_{\alpha,\gamma}| \tag{55}$$

$$WS^D = \lambda_E E^D_{MAE} + \lambda_F F^D_{MAE} + \lambda_S S^D_{MAE}. \tag{56}$$

Here all of the variables and indices are identical to those contained in the previously defined MSE and loss function equations.

E. List of Allegro models generated and evaluated in this study

TABLE II. List of 10 different Allegro models studied in this work. Note that the labels serve as a compact notation for the model definition, in the format XYZ, where X is the input dataset label (C for "combined", S for "single"), Y is the model size label (S for "small", M for "medium" as defined in Table I), and Z is simple a numerical label for further model differentiation, such as different model initialization.

Label	Model Size	Training Structure(s)
CS1	Small	S1 + S2
CS2	Small	S1 + S2
CS3	Small	S1 + S2
CS4	Small	S1 + S2
CS5	Small	S1 + S2
CS6	Small	S1 + S2
CM1	Medium	S1 + S2
CM2	Medium	S1 + S2
SS1	Small	S1
SS2	Small	S2

Based on the ideas presented earlier in this section, a total of 10 Allegro models were trained and tested in the present work. These models incorporate testing of "small" and "medium" model sizes, of training on structures $S1$ and $S2$ alone and combined, and of the variability between similar models when initialized differently prior to optimization. As mentioned in Sec. IV B, the initialization of the model is dependent on two random seeds: the seed which defines the random initialization of the learned parameters in the network, and the seed which defines which input configurations are split into training and validation sets. The differences in initialization of otherwise identical models impacts the model's sources of learning and subsequent optimization pathways, adding a source of "noise" to the process. The ten models assessed are detailed in Table II, where each model has a different initialization.

V. MOLECULAR DYNAMICS DETAILS

A. Molecular dynamics simulations representing micro-canonical ensembles

While there are many ways to perform molecular dynamics simulations, the present work focuses on simulations based on the micro-cannonical (NVE) ensemble, in which the number of particles (ions) N_I, the volume Ω and the total energy E (meaning the sum of the kinetic energy and interaction energy) are kept constant. In this case, the temperature varies in time t such that $T(t)$ fluctuates over a well controlled range. Performing the simulations using NVE ensembles enables the study of numerical properties of the simulations with some reproducibility. In terms of comparison with experiment, a disadvantage of using an NVE ensemble is that the volume rigidity results in relatively large pressures within the system during the molecular dynamics time steps, likely exceeding the experimental conditions. On the other hand, for a very long ($\sim$1 ns) molecular dynamics simulation using an NVE ensemble as implemented in LAMMPS , we find that with fixed input data for more than one run, we can obtain identical trajectories using a given computer CPU hardware configuration. In the present work, this reproducibility is very helpful to the analysis of the complicated results. However, we note that trajectories start to vary after approximately 1 ps of simulation time when using different CPU hardware architectures and results from runs using GPU hardware for the molecular dynamics calculations generally differ on each run. This sensitivity of the trajectories to the computer hardware is direct evidence of the Lyapunov instability discussed in Sec. V B. In practice, the molecular dynamics runs using the LAMMPS software package [16] uses the "Velocity-Verlet" [61–64] algorithm to perform the time integration.

As an example of NVE simulation stability, we show plots of the variation of temperatures and total energies throughout a molecular dynamics run using Allegro model CS2 and LAMMPS initial velocity distribution at target temperatures 350K, 450K, 550K, and 900K for structure S1 in Fig. 3 and structure S2 in Fig. 4. We see that in both examples, the temperature is reasonably stable at roughly $\pm$ 100K, while the total energy is much better controlled, with deviations in temperature units of only approximately $\pm$ 20K.

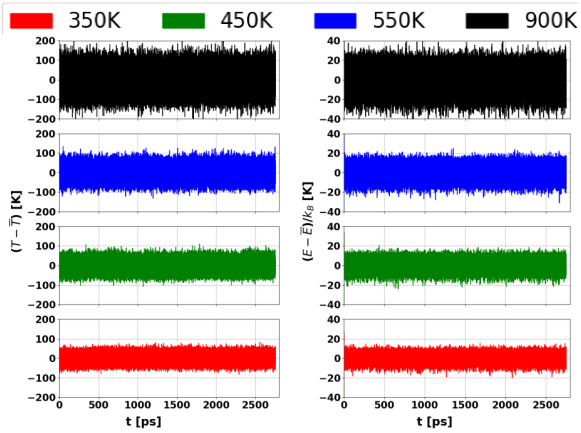


FIG. 3. Plot of deviations of temperature (left) and total energy (right) from their respective averages for Allegro model CS2 during molecular dynamics runs at target temperature 350K, 450K, 550K, and 900K, initialzed with reference structure S1. Note that there is a 5-fold difference in the vertical scales for the two plots.

B. Behavior of long molecular dynamics simulations for well behaved potential models

In principle, the ionic positions $\mathbf{r}_i(t)$ discussed in Section III A can be well-characterized from molecular dynamics simulations, whether the interaction potential of the system is evaluated using model potentials, density functional theory, or by using machine learned interatomic potentials. The use of

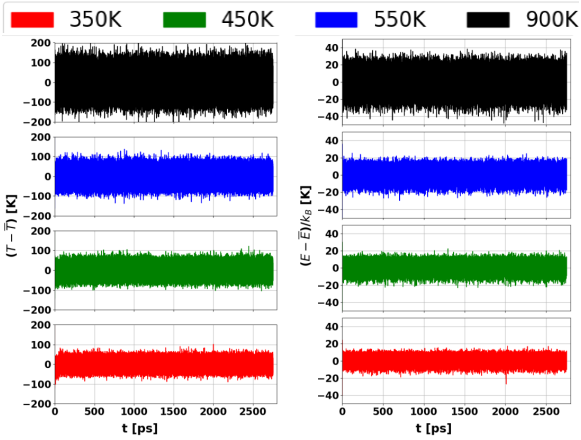


FIG. 4. Plot of results analogous to those presented in Fig. 3 for a molecular dynamics run initialized with reference structure S2.

machine learned interatomic potentials enables the molecular dynamics simulation times to extend into the nanosecond range, exposing issues associated with very long molecular dynamics results.

In fact, much has been written about divergent results in long molecular dynamics simulations. For example, Frenkel and Smit [34] note that molecular dynamics trajectories of particles involved with complicated inter-particle interactions are very sensitive to the computational details including the initial conditions and the computer hardware and software used in the simulation; a phenomenon known as the "Lyapunov instability". [65]. Lyapunov, is known as the founder of modern stability theory based on his work in the late 1800's. In the context of molecular dynamics, the focus is on finding particle trajectories as a function of time. It may be the case that after some time t', the trajectories at times $t > t'$ are "shifted" in the sense that the numerical integration of those trajectories are no longer numerically derivable from the earlier times. (Note that the Lyapunov instabilities have exponential time divergences so that a t' measure is qualitative.) This extreme sensitivity to time conditions is well known for partitial differential equation having nonlinear spatial terms. The instability of the particle trajectories occurs even when the numerical algorithm used in the simulation well maintains the conserved quantities of the simulation (in our case NVE). Evidence of this effect can be observed when performing simulations with reproducible algorithms in terms of the sensitivity of the produced ion trajectories to the computer hardware used. Frenkel and Smit suggest that this phenomenon can be generally helpful for performing ensemble averages, since the "shifted" trajectories represent physically reasonable configurations of the system. However, when the causal relationship between $\widetilde{\mathbf{P}}_D(\tau)$ and $\widetilde{\mathbf{P}}_D(0)$, is no longer meaningful due to the Lyapunov instability, their use in evaluation in the Green-Kubo Eq. (9) becomes problematic.

To our knowledge, the ideal averaging methodology that can take advantage of the Lyapunov instability is not yet known. However, we are aware of some literature discussions on this topic. For example, in 1989, Posch and Hoover [66] discussed ways in which knowledge of the "Lyapunov spectrum" might be used within molecular dynamics simulations. In 2009, Skeel [67] published a paper entitled "What makes molecular dynamics work?" which suggests that the results of molecular dynamics simulations should be interpreted in the statistical sense. From the statistical viewpoint, it is possible to develops algorithms to use molecular dynamics results to estimate some measureable quantities of interest, but with the statistical viewpoint, the meaning of time sequences of the trajectories $\{\mathbf{r}_i(t)\}$ is altered. More recently, there have been a number of papers in the literature, which don't necessarily directly mention the notion of Lyapunov instability, but develop methods of estimating, analyzing, or correcting for "noise" in molecular dyanamics simulations such as in references [68–72]. Because of the mathematics of non-linear partial differential equations, chaotic behavior of physical phenomena is found in a wide range of physics disciplines. A general review of some of these and how the chaotic behavior can be affected by other physical phenomena such as dissipative effects is found for example in a review article by Krechetnikov and Marsden [73]. In the present work, we use the methods described in Sec. III B to analyze the results of molecular dynamic simulations without explicit assumptions of Lyapunov instabilities or other noise effects encountered in our focused study of the Green-Kubo ionic conductivity. In this way, it is possible to observe some of the effects of Lyapunov instabilities under somewhat controlled conditions. We observe that they do not always completely dominate evaluations of equations of motion as illustrated in the results Sec. VI C based on very long simulations using the well-behaved Allegro models, particularly when representing lower temperature environments.

C. Anomolous behavior of molecular dynamics runs for some machine learned models

In contrast to the well-controlled temperature behavior $T(t)$ shown in Figs. 3 and 4 , we occasionally found that during a long molecular dynamics run, some of the machine learned models failed to represent physically reasonable configurations, particularly at higher temperatures. This behavior has been previously reported in the literature such as in the recent paper by Yan and co-workers [60].

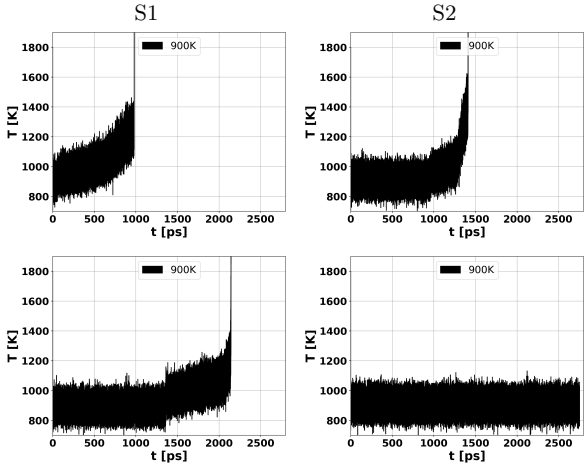


FIG. 5. Plots (S1) and (S2) show the temperatures $T(t)$ (in K) for molecular dynamics simulations using LAMMPS with the Allegro model CS1 at target temperature 900K as a function of simulation time (in ps units) for structures S1 and S2, respectively. Each simulation used a different velocity distribution.

In the present work, we screened each of the Allegro models using long molecular dynamics runs at the high target temperture of 900K with two different initial velocity distributions. Of the 10 constructed Allegro models, only 2 were found to have divergent behavior from the molecular dynamics screening test. For example, the divergent behavior was found for Allegro model CS1 as evidenced in the temperature profile plots shown in Fig. 5. In this case, the temperature plots show completely different profiles depending on the reference structure and the initial velocity distribution. In addition, set CM2 also failed with a divergent temperature profile, although in this case it was only a single simulation for one of the structures which failed. A likely explanation for this behavior is that during the course of the run, configurations were accessed that were not well-enough represented in the training data. As discussed in Sec. IV C, adding extra training configurations that include sufficiently small atomic separations qualitatively seems to reduce the number of Allegro models which fail the molecular dynamics screen test. We note that all 10 constructed Allegro models could be used to examine the minimum energies and volumes as discussed in Sec. VI B.

VI. RESULTS AND ANALYSIS

A. Analysis of Allegro model training

This section includes the training results for the 10 Allegro models tested, which are described in Sec. IV E. These results include some details regarding the models' stopping behavior in Table III, which is discussed in Sec. IV B and the errors for the models' training and validation set defined by Allegro's internal metrics, as discussed in Sec. IV D, which are shown in Tables IV and V.

TABLE III. Table detailing the training behavior of the 10 Allegro models investigated, including the total number of epochs trained, the epoch of the "best" model, and the stopping behavior of the training.

Model	Best Epoch	Final Epoch	Stopping Type
CS1	160	660	plateau
CS2	11337	11401	timeout
CS3	8789	9123	timeout
CS4	798	1298	plateau
CS5	8220	8720	plateau
CS6	142	642	plateau
CM1	10924	10934	timeout
CM2	7967	8053	plateau
SS1	11955	12399	timeout
SS2	12426	12465	timeout

Since all of the models have different initializations, we expect that they will have different local minima that they are optimizing towards. This can be seen clearly from the differences between the models' training behavior shown in Table III. Here we see that they all train for differing lengths of time before stopping, and stop under different criteria: some approach the minimum and optimization stagnates, resulting in a training plateau, while others progress towards a shallow minimum and continue to incrementally improve before simply running out of time. It is important to note that while the models that ran out of time are still technically improving, in our experience this improvement is negligible and

TABLE IV. Table detailing the internal Allegro error metrics for the training dataset of the 10 Allegro models investigated. These metrics include the WS ($\times 10^{-3}$), E_{MAE} (meV), F_{MAE} (meV/Å), and S_{MAE} (meV/Å^3).

Model	WS^{train}	E^{train}_{MAE}	F^{train}_{MAE}	S^{train}_{MAE}
CS1	284.52	834.94	18.10	0.50
CS2	18.31	43.92	10.84	0.18
CS3	16.16	37.88	10.40	0.19
CS4	115.67	332.15	14.50	0.37
CS5	14.46	32.78	10.41	0.18
CS6	319.92	940.03	19.07	0.65
CM1	9.84	24.61	4.79	0.12
CM2	9.58	23.66	4.96	0.12
SS1	25.78	65.91	11.21	0.22
SS2	21.70	53.92	10.99	0.19

TABLE V. Table detailing the internal Allegro error metrics for the validation dataset of the 10 Allegro models investigated. These metrics include the WS ($\times 10^{-3}$), E_{MAE} (meV), F_{MAE} (meV/Å), and S_{MAE} (meV/Å^3).

Model	WS^{val}	E^{val}_{MAE}	F^{val}_{MAE}	S^{val}_{MAE}
CS1	13.28	22.51	17.00	0.34
CS2	8.29	14.01	10.67	0.17
CS3	8.27	14.13	10.51	0.17
CS4	11.92	21.71	13.79	0.25
CS5	8.09	13.71	10.41	0.17
CS6	13.80	23.18	17.70	0.54
CM1	3.64	5.89	4.92	0.10
CM2	3.86	6.39	5.08	0.10
SS1	9.02	15.48	11.41	0.18
SS2	8.60	14.68	10.97	0.17

doesn't justify continuing training past two weeks for a slightly "better" model. We also note that the models will have different training and validation set selections from the input dataset because of the different initialization seeds, resulting in different configuration spaces where the model is accurate and different interpolative capabilities as well.

Upon analysis of the internal Allegro errors presented in tables IV and V, we see that the small models all have similar training and validation weighted sum errors, on the order of 10 meV, except for the outliers CS1 and CS6. These have an order of magnitude larger training weighted sum errors on the order of 100 meV, which is the result of relatively large energy E_{MAE} errors. We note that these models both trained for a very short time of around 150 epochs, much shorter than all other models, yet they both stopped due the optimization plateauing, meaning the models were no longer improving. This suggests that the initialization of these models may have started near a very high minimum with less room for improvement in errors as training progressed. This demonstrates the importance of model initialization by showing how similar models can be quite different due to their initialization conditions alone. Of these two small models with higher error, one failed the 900K stability test while the other passed, suggesting higher model error is not necessarily indicative of a model's stability during molecular dynamics simulations.

The medium models were found to have slightly lower errors overall compared to the small models, but only minimally so - a difference of at most a factor of two. Interestingly, the single structure small models did have slightly higher training errors compared to the combined structure small models (ignoring the outliers CS1 and CS6), but have similar validation errors. We expect that this difference is not meaningful since they have significantly different training and validation configurations.

B. Analysis of LAMMPS minimize calculations

In addition to the internal Allegro measures of error, it is important to examine the performance of the model interatomic potentials directly. In this section, we examine ground state structure prediction from LAMMPS minimization calculations for the 10 Allegro models investigated. For each Allegro model these calculations involve initializing with one of the two reference structures S1 or S2 and then allowing the LAMMPS "minimize" command to find a nearby optimized structure. The optimized structure can then be analyzed in terms of its energy difference ΔE and its volume difference $\Delta\Omega$ relative to the corresponding reference structure. Since the ground state structures are not included in the training or validation datasets for any of the models, this serves as a measure of interpolation to a specific configuration. It is important that the ground state structures are fairly well represented, as they are used as starting configurations when running equilibrium based molecular dynamics. If the minimization calculations result in significant volume and/or energy differences, then the models are not representing the ground state structures as accurately.

As we can see in Table VI, the energy changes vary widely across models, in the range of 10^{-1} to 10^{1} meV, while the volume changes vary less extremely, in the range of 10^{-2} to 10^{0} Å^3. Although this is a very limited test of a model's ability to interpolate, we're interested to see if these results are indicated by the validation set's weighted sum for the model –

TABLE VI. Table showing LAMMPS minimzation results for all 10 Allegro models. Volumes are given in units of $\mathring{A}^3$ and energies in meV. WS values are given $\times 10^{-3}$. Entries in red denote calculations performed on starting structures not trained by the model.

Model	WS^{val}	$\Delta\Omega_{S1}$	$\Delta\Omega_{S2}$	ΔE_{S1}	ΔE_{S2}
CS1	13.28	−0.99	−0.56	−11.03	+18.98
CS2	8.29	−0.39	+0.33	−3.46	−4.82
CS3	8.27	−0.37	+0.23	−7.53	−1.99
CS4	11.92	−1.09	−0.40	−36.57	−5.15
CS5	8.09	−0.05	+0.09	−8.70	+0.67
CS6	13.80	−1.30	−0.38	−0.42	+15.62
CM1	3.64	−0.13	−0.26	+1.60	+1.09
CM2	3.86	−0.02	−0.40	+1.12	+1.45
SS1	9.02	−0.37	+0.75	−10.54	−53.14
SS2	8.60	+0.01	−0.28	+25.24	−5.83

the benchmark by which we are selecting the "best" model – based on the assumption that it is a good measure of the model's ability to interpolate well to untrained structures.

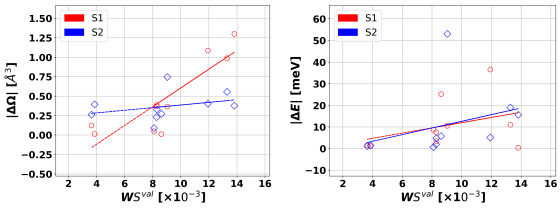


FIG. 6. Graphical representations of volume errors $\Delta\Omega$ (left) and energy errors ΔE (right) for all 10 Allegro models listed in Table VI represented with open circles versus the validation error WS, with structure S1 and S2 given by red and black colors, respectively. The straight lines indicate trends in terms linear least squares fits.

Figure 6 shows plots of the volume change vs validation weighted sum and energy change vs validation weighted sum for each of the 10 Allegro models. We see that for both plots there is an observable trend between the weighted sum of the validation dataset and the model's ability to correctly represent the ground state structure. We note that the outliers which have large $\Delta\Omega$ and ΔE in spite of having lower validation error come from the models that are trained on a single structure being applied to the untrained structure. Since it should not necessarily be the case that a model can interpolate to configurations generated from untrained structures, this behavior is expected.

C. Analysis of LAMMPS molecular dyanamics simulations based on ALLEGRO trained models

In this section, we analyze the results of the LAMMPS simulations for the Allegro models that were tested and deemed stable based on the 900K stability screening discussed in Sec. V C. From this screening, only 2 models failed - CS1 and CM2, which were not analyzed further. We also omit model CM1 from further analysis due to computational resource limitations and time constraints. In Sec. VI C 1 we focus on analyzing the molecular dyanamics results of 5 small models (CS2-CS6) trained with combined structure S1 and S2 configurations. In Sec. VI C 2 we discuss the remaining 2 models (SS1 and SS2) which were trained with only structure S1 or S2 configurations.

1. *Analysis of small CS2 - CS6 models trained on combined S1 and S2 configurations*

The 5 small models, CS2 - CS6, were used to run molecular dynamics simulations in LAMMPS at target temperatures of 350K, 450K, and 550K. For each temperature, 10 different simulations with different initial velocity configurations were run for 2.8 *ns*. As discussed in Secs. III B 1 and III B 2, we consider averaging over not only the block intervals of a single simulation, but also over multiple simulations with different starting velocity distributions – and by extension, different trajectories – for each model.

Here we present the results for model CS2, where we show block averaged $\overline{\Lambda}^{\chi}_{\Phi v}$ plots for each of the 10 simulations as well as the $\overline{\overline{\Lambda}}^{\chi}_{\Phi}$ plots additionally averaged over the 10 simulations. In Figs. 7 and 8, we see the full and tracer results for structure S1, and in Figs 9 and 10, we see the full and tracer results for structure S2. Similar plots for the remaining models are given in the Supplemental Materials [27].

These plots demonstrate the wide range of resultant interval function behavior over time interval τ for the different simulations and show that block averaging alone is not sufficient, particularly at the 550K target temperature. While the tracer approximation curves are fairly well controlled, the full conductivity curves are not not - they are quite widely spread out and don't seem to be exhibiting the expected asymptotic behavior in the long τ limit.

Conversely, if we additionally average over the simulations with different initial velocity configurations, we see much better asymptotic behavior for

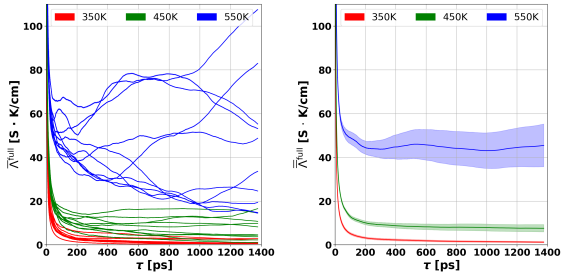


FIG. 7. Left plot shows $\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)$ versus time interval τ for structure S1 for Φ corresponding to Allegro model CS2 evaluated at at target temperatures 350K, 450K, and 550K each for 10 different initial velocity distributions. The standard error $\Delta\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)$ is indicated with light colored shading and is very small. Right plot shows the corresponding results for $\overline{\overline{\Lambda}}^{\rm full}_{\Phi}(\tau)$ with a bold line and the standard error $\Delta\overline{\overline{\Lambda}}^{\rm full}_{\Phi}(\tau)$ with light colored shading at target temperatures 350K, 450K, and 550K.

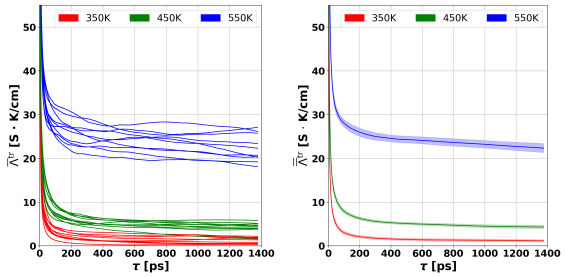


FIG. 8. Left plot shows $\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau)$ versus time interval τ for structure S1 for Φ corresponding to Allegro model CS2 evaluated at at target temperatures 350K, 450K, and 550K each for 10 different initial velocity distributions. The standard error $\Delta\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau)$ is indicated with light colored shading and is very small. Right plot shows the corresponding results for $\overline{\overline{\Lambda}}^{\rm tr}_{\Phi}(\tau)$ with a bold line and the standard error $\Delta\overline{\overline{\Lambda}}^{\rm tr}_{\Phi}(\tau)$ with light colored shading at target temperatures 350K, 450K, and 550K.

the full conductivity interval functions. Additionally, we see that higher temperatures in general are less well controlled than lower temperatures.

As discussed in Sec. III B 3, we can also define a triple averaged $\overline{\overline{\overline{\Lambda}}}^{\chi}$ function by additionally averaging over similar Allegro models. Here we present the results for the doubly averaged $\overline{\overline{\Lambda}}^{\chi}_{\Phi}$ plots for all 5 models, as well as the $\overline{\overline{\overline{\Lambda}}}^{\chi}$ plots additionally averaged over the models. In Figs. 11 and 12, we

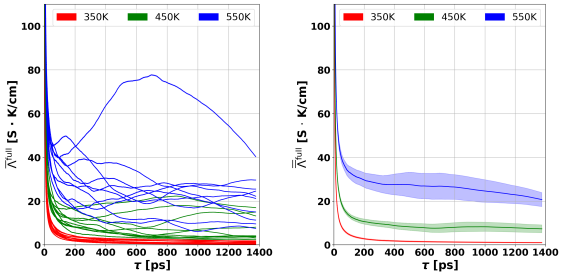


FIG. 9. Plots analogous to those presented in Fig. 7 evaluated for structure S2.

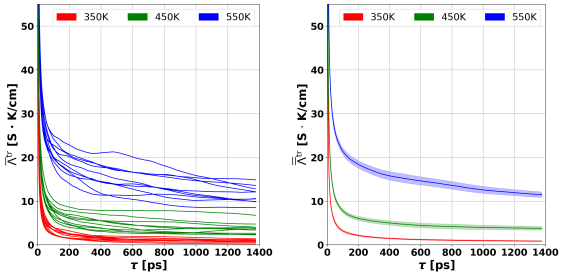


FIG. 10. Plots analogous to those presented in Fig. 8 evaluated for structure S2.

see the full and tracer results for structure S1. In Figs. 13 and 14, we see the full and tracer results for structure S2. We see that the triple averaged interval function curves are similarly well behaved with expected asymptotic behavior at large τ, with somewhat reduced errors compared to the doubly averaged interval function curves. Similar plots for the remaining models are given in the supplemental materials [27].

Given the doubly averaged $\overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau)$ and triply averaged $\overline{\overline{\overline{\Lambda}}}^{\chi}(\tau)$ plots, we can perform a least squares regression in order to take the limit as τ goes to infinity and estimate $\overline{\overline{\sigma T}}\rfloor^{\chi}_{\Phi}$ for each model as well as $\overline{\overline{\overline{\sigma T}}}\rfloor^{\chi}$ for the model average, as mentioned in Sec. III B 4. In Fig.s 15 and 16, we see examples of fits to $\overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau)$ for Allegro model CS2 and $\overline{\overline{\overline{\Lambda}}}^{\chi}(\tau)$ for the 5 Allegro models for both the full and tracer data for structure S1. In Fig.s 17 and 18, we see examples of fits to $\overline{\overline{\Lambda}}^{\chi}_{\Phi}(\tau)$ for Allegro model CS2 and $\overline{\overline{\overline{\Lambda}}}^{\chi}(\tau)$ for the 5 Allegro models for both the full and tracer data for structure S2. Similar plots for the remaining models are given in the supplemental materials [27].

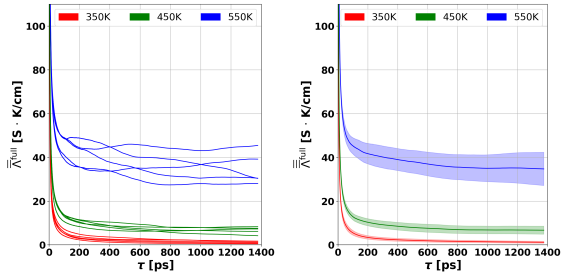


FIG. 11. Left plots show $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ as a function of time interval τ for structure S1 at target temperatures 350K, 450K, and 550K each for Φ corresponding to Allegro models CS2, CS3, CS4, CS5, and CS6. The standard error is not indicated in these plots. Right plot shows the corresponding results for the triple averaged interval function $\overline{\overline{\Lambda}}^{\text{full}}(\tau)$ with a bold line and the standard error $\Delta\overline{\overline{\Lambda}}^{\text{full}}(\tau)$ with light colored shading at target temperatures 350K, 450K, and 550K.

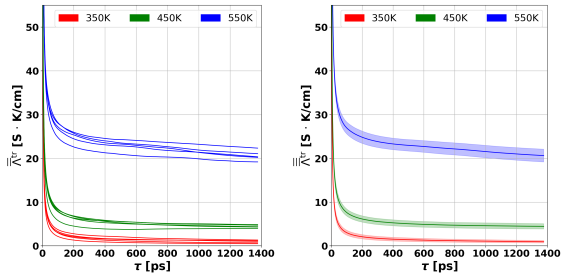


FIG. 12. Left plots show $\overline{\overline{\Lambda}}^{\text{tr}}_{\Phi}(\tau)$ as a function of time interval τ for structure S1 at target temperatures 350K, 450K, and 550K each for Φ corresponding to Allegro models CS2, CS3, CS4, CS5, and CS6. The standard error is not indicated in these plots. Right plot shows the corresponding results for the triple averaged interval function $\overline{\overline{\Lambda}}^{\text{tr}}(\tau)$ with a bold line and the standard error $\Delta\overline{\overline{\Lambda}}^{\text{tr}}(\tau)$ with light colored shading at target temperatures 350K, 450K, and 550K.

Here we see that for the 550K interval function data, which is less well controlled overall and often has issues with asymptotic behaviour at long τ, the fits to the data are less precise as a result, often deviating significantly from the expected asymptotic value at long τ when visually inspecting the data. In contrast, the lower temperatures, which are much better controlled and have better asymptotic behavior, generally have better fits as a result.

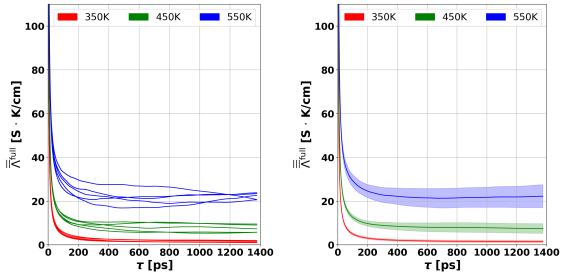


FIG. 13. Plots analogous to Fig. 11 for reference structure S2.

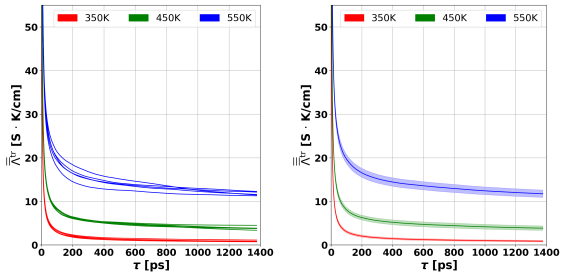


FIG. 14. Plots analogous to Fig. 12 for reference structure S2.

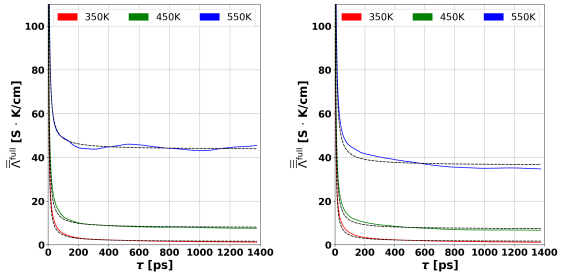


FIG. 15. Plots of the analytic fitting function defined in Eq. (33) indicated by the dashed lines for target temperatures 350K, 450K, and 550K. On the left, the fit and the function correspond to the doubly averaged interval function $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ with Φ represented Allegro model CS2. On the right, the fit and the function correspond to the triply averaged interval function $\overline{\overline{\Lambda}}^{\text{full}}(\tau)$. The evaluations were based on reference struction S1.

The $\overline{\overline{\sigma T}}\rfloor^{\chi}_{\Phi}$ estimates for each of the 5 small Allegro models, as well as the $\overline{\overline{\overline{\sigma T}}}\rfloor^{\chi}$ estimates for the model average across the 5 models were analyzed for structures S1 and S2. The tables with this data are given in the supplemental materials. From these re-

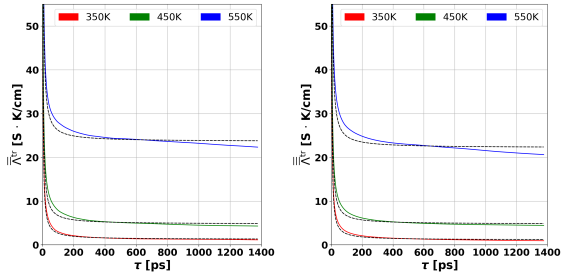


FIG. 16. Plots analogous to Fig. 15 but evaluated for the tracer interval functions $\overline{\overline{\Lambda}}^{\mathrm{tr}}_{\Phi}(\tau)$ and $\overline{\overline{\Lambda}}^{\mathrm{tr}}(\tau)$.

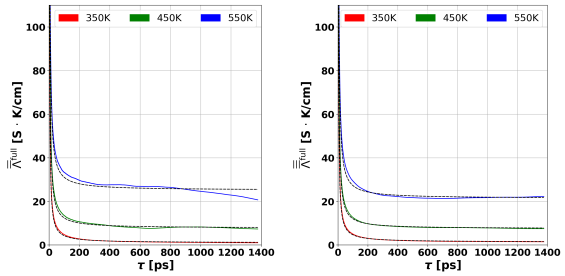


FIG. 17. Plots analogous to Fig. 15 but evaluated for reference structure S2.

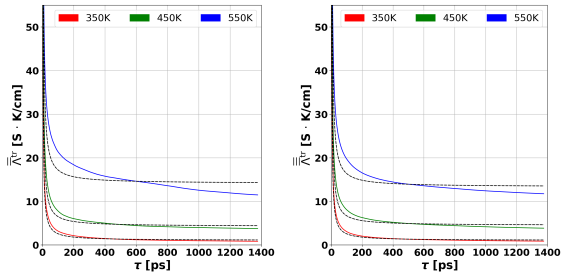


FIG. 18. Plots analogous to Fig. 16 but evaluated for reference structure S2.

sults, we can estimate conductivity by dividing by the doubly averaged temperature, $\overline{\overline{T}}_{\Phi}$ for the individual models, or $\overline{\overline{\overline{T}}}$ for the model average. These conductivity estimates can then be used to estimate the Haven ratio, as discussed in Sec. III A. The full and tracer conductivity estimates along with the Haven ratio estimates are given in Tables VII for structure S1 and VIII for structure S2.

From the estimated conductivity values in tables VII and VIII, we can construct Arrhenius plots for the tracer and full conductivity results as discussed in Sec. III B 4 for each of the 5 small Allegro models, which is shown in Fig. 19 for structure S1 and Fig. 20 for structure S2, as well as the model average for both structures S1 and S2, which is shown in Fig. 21.

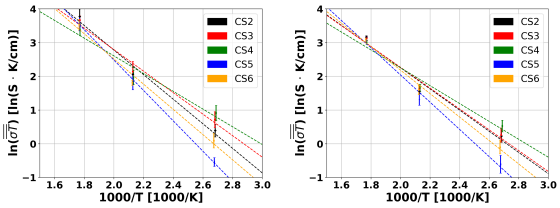


FIG. 19. Arrhenius plots of full (left) and tracer (right) conductivity results for each of the Allegro models CS2 - CS6 for structure S1.

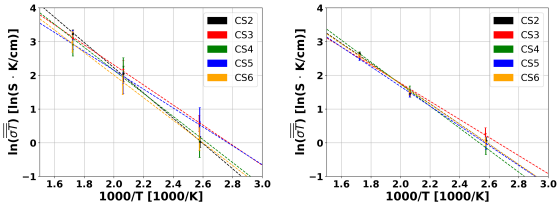


FIG. 20. Arrhenius plots of full (left) and tracer (right) conductivity results for each of the Allegro models CS2 - CS6 for structure S2.

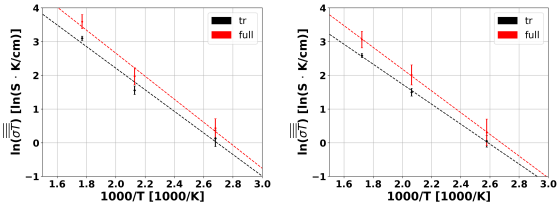


FIG. 21. Arrhenius plots of full and tracer conductivity results for the CS2 - CS6 model average for structure S1 (left) and S2 (right).

We see that that the slopes in the Arrhenius plots vary significantly, depending on the Allegro model, in part due to the small number of target temperatures in this study. We also note that the 550 K target temperature results are less precise due to the Lyapunov effects as illustrated in the $\overline{\Lambda}^{\chi}_{\Phi v}(\tau)$ plots, which also contribute to the slope variations in the Arrhenius plots. Consequently, estimates of E_a are similarly imprecise. For illustrative purposes, estimates of E_a are listed in Table IX for each structure

TABLE VII. Table detailing the $\overline{\overline{\sigma}}$ results for each of the stable small models investigated from LAMMPS molecular dynamics utilizing starting structure S1 at target temperatures of 350K, 450K, and 550K. The model average $\overline{\overline{\sigma}}$ is also included under the "AVG" label. Haven ratios H_R are calculated for each model, as well as the model average. Here temperatures are given in K, conductivity results are given in mS/cm, and H_R results are unitless.

Model	Struct	T^{targ}	$\overline{\overline{T}}_\Phi$	$\overline{\overline{\sigma}}_\Phi^{\text{tr}}$	$\overline{\overline{\sigma}}_\Phi^{\text{full}}$	H_R
CS2	S1	350K	372.84 ± 0.32	3.38 ± 0.62	3.98 ± 0.75	0.85 ± 0.22
CS3	S1	350K	373.13 ± 0.35	3.49 ± 0.73	5.49 ± 1.31	0.64 ± 0.20
CS4	S1	350K	372.04 ± 0.46	4.56 ± 0.80	6.70 ± 1.51	0.68 ± 0.19
CS5	S1	350K	373.64 ± 0.32	1.46 ± 0.38	1.56 ± 0.20	0.94 ± 0.27
CS6	S1	350K	374.09 ± 0.34	2.43 ± 0.48	2.93 ± 0.63	0.83 ± 0.24
AVG	S1	350K	373.15 ± 0.50	3.06 ± 0.74	4.13 ± 1.18	0.74 ± 0.28
CS2	S1	450K	470.09 ± 0.17	10.12 ± 0.76	16.70 ± 3.55	0.61 ± 0.14
CS3	S1	450K	469.99 ± 0.15	10.83 ± 1.25	18.23 ± 5.45	0.59 ± 0.19
CS4	S1	450K	469.15 ± 0.15	11.18 ± 1.27	15.30 ± 3.30	0.73 ± 0.18
CS5	S1	450K	470.09 ± 0.20	7.89 ± 1.31	13.42 ± 3.15	0.59 ± 0.17
CS6	S1	450K	470.75 ± 0.16	10.51 ± 1.46	13.02 ± 1.01	0.81 ± 0.13
AVG	S1	450K	470.01 ± 0.30	10.11 ± 1.28	15.33 ± 3.89	0.66 ± 0.19
CS2	S1	550K	565.45 ± 0.10	41.76 ± 1.76	77.11 ± 17.25	0.54 ± 0.12
CS3	S1	550K	565.17 ± 0.11	40.34 ± 2.62	69.28 ± 8.72	0.58 ± 0.08
CS4	S1	550K	564.65 ± 0.12	35.95 ± 2.35	52.19 ± 9.32	0.69 ± 0.13
CS5	S1	550K	565.28 ± 0.13	39.38 ± 3.12	62.29 ± 13.35	0.63 ± 0.14
CS6	S1	550K	566.34 ± 0.17	38.79 ± 1.93	60.40 ± 9.84	0.64 ± 0.11
AVG	S1	550K	565.38 ± 0.30	39.24 ± 2.57	64.25 ± 13.45	0.61 ± 0.13

TABLE VIII. Table detailing the $\overline{\overline{\sigma}}$ results for each of the stable small models investigated from LAMMPS molecular dynamics utilizing starting structure S2 at target temperatures of 350K, 450K, and 550K. The model average $\overline{\overline{\sigma}}$ is also included under the "AVG" label. Haven ratios H_R are calculated for each model, as well as the model average. Here temperatures are given in K, conductivity results are given in mS/cm, and H_R results are unitless.

Model	Struct	T^{targ}	$\overline{\overline{T}}_\Phi$	$\overline{\overline{\sigma}}_\Phi^{\text{tr}}$	$\overline{\overline{\sigma}}_\Phi^{\text{full}}$	H_R
CS2	S2	350K	387.54 ± 0.39	2.74 ± 0.28	2.65 ± 0.44	1.03 ± 0.20
CS3	S2	350K	388.41 ± 0.06	3.37 ± 0.59	4.59 ± 1.09	0.74 ± 0.22
CS4	S2	350K	388.04 ± 0.26	2.20 ± 0.42	2.72 ± 1.33	0.81 ± 0.43
CS5	S2	350K	387.89 ± 0.51	2.54 ± 0.47	4.77 ± 2.07	0.53 ± 0.25
CS6	S2	350K	388.27 ± 0.65	2.69 ± 0.34	2.89 ± 0.98	0.93 ± 0.34
AVG	S2	350K	388.03 ± 0.45	2.71 ± 0.47	3.52 ± 1.38	0.77 ± 0.33
CS2	S2	450K	484.57 ± 0.11	8.84 ± 0.85	16.00 ± 3.68	0.55 ± 0.14
CS3	S2	450K	484.86 ± 0.11	9.14 ± 0.78	18.21 ± 4.77	0.50 ± 0.14
CS4	S2	450K	484.89 ± 0.10	9.89 ± 1.19	19.34 ± 5.49	0.51 ± 0.16
CS5	S2	450K	485.19 ± 0.13	9.02 ± 1.09	12.10 ± 3.77	0.75 ± 0.25
CS6	S2	450K	485.55 ± 0.11	9.45 ± 0.94	11.51 ± 3.34	0.82 ± 0.25
AVG	S2	450K	485.01 ± 0.20	9.27 ± 1.05	15.43 ± 4.58	0.60 ± 0.19
CS2	S2	550K	580.41 ± 0.08	24.31 ± 1.06	43.28 ± 5.52	0.56 ± 0.08
CS3	S2	550K	580.68 ± 0.09	22.72 ± 1.04	38.25 ± 6.70	0.59 ± 0.11
CS4	S2	550K	580.40 ± 0.09	23.75 ± 1.85	31.61 ± 10.57	0.75 ± 0.26
CS5	S2	550K	580.79 ± 0.08	20.89 ± 1.13	36.52 ± 10.17	0.57 ± 0.16
CS6	S2	550K	581.48 ± 0.08	23.14 ± 1.65	34.32 ± 10.40	0.67 ± 0.21
AVG	S2	550K	580.75 ± 0.22	22.96 ± 1.42	36.80 ± 9.00	0.62 ± 0.16

S1 and S2 for each of the 5 small Allegro models and their average, as discussed in Sec. III B 4.

TABLE IX. Table containing the activation energy calculations for structures S1 and S2 based on the previous full and tracer conductivity estimations. Here "AVG" denotes the results from the triple averaged conductivity estimates. Energy units are given in meV.

Model	Struct	E_a^{tr}	E_a^{full}
CS2	S1	270.52 ± 48.53	313.75 ± 42.42
CS3	S1	266.00 ± 40.40	274.50 ± 39.60
CS4	S1	228.31 ± 40.61	226.34 ± 47.88
CS5	S1	347.75 ± 36.06	388.03 ± 13.83
CS6	S1	299.60 ± 25.53	321.95 ± 39.43
AVG	S1	275.65 ± 40.51	293.97 ± 41.32
CS2	S2	257.95 ± 19.02	322.47 ± 13.09
CS3	S2	231.16 ± 19.93	256.21 ± 13.51
CS4	S2	281.09 ± 7.66	294.24 ± 55.29
CS5	S2	252.20 ± 5.64	241.11 ± 38.99
CS6	S2	256.73 ± 7.97	288.16 ± 19.75
AVG	S2	254.94 ± 10.05	277.58 ± 13.53

From analysis of the model average full conductivity estimates, we find that structure S1 has a conductivity of 4.13 mS/cm at 373.15 K and structure S2 has a conductivity of 3.52 mS/cm at 388.03 K, which seem to be quite reasonable results. Overall we find that structure S1 has higher conductivity than structure S2. We also see that in general, the Haven ratio decreases with increasing temperature and is less than 1.0, indicating appreciable ionic correlations in the material. This highlights the fact that the tracer conductivity alone is not sufficient and that calculating the full conductivity is necessary to more accurately estimate the true conductivity of the material. From the full conductivity Arrhenius results, we find the average activation energy for structure S1 to be 293.97 meV and 277.58 meV for structure S2.

When assessing the errors in the reported conductivity estimates, we note that for both structure S1 and S2, the model average error increases as temperature increases, and that the full conductivity errors are larger than their tracer error counterparts. For structure S1 at target temperatures of 350K and 450K, the model average full conductivity errors are 1.18 mS/cm and 3.89 mS/cm. For structure S1 at 550K target temperature, on the other hand, has a much larger model average full conductivity error of 13.45 mS/cm. This shows that the full conductivity at 550K is not well represented due to the poorly controlled interval function behavior mentioned previously. For structure S2, the pattern holds, where the model average full conductivity error at 550K target temperature is 9.00 mS/cm, compared to the 1.38 mS/cm and 4.58 mS/cm errors at target temperatures of 350K and 450K. We see that the majority of the error is coming from the Lyapunov instability present when averaging over multiple simulations with different initial velocity distributions, and that the block averaging error for an individual simulation is insignificant. The error associated with averaging over multiple Allegro models is not trivial, but is small compared to the dominant error from the velocity distribution averaging. Although the activation energy estimates have issues, we note that their errors are reasonable, 41.32 meV for structure S1 and 13.53 meV for structure S2 when derived from the model average full conductivity data.

Given that our full conductivity estimates still have somewhat high errors, and the fact that our model structures S1 and S2 were derived though a simplification and idealization of the work of Deng and collaborators [23], we do not expect our conductivity results to numerically agree with their experimental or computational results. The full conductivities we found at temperatures of roughly 565K for structure S1 and 580K for structure S2 were around 10 times larger than those determined by Deng and collaborators at a lower temperature near 350K. Similarly, the activation energies estimated in this work are roughly 100 meV smaller than those of Deng and collaborators.

2. *Analysis of small SS1 and SS2 models trained on only S1 or S2 configurations*

The small SS1 and SS2 models, which were trained on only S1 and S2 structural configurations respectively, were used to run molecular dynamics simulations with the same setup as the small CS2 - CS6 models described in the previous section: 10 simulations with different initial velocity distributions were ran at each of the 3 target temperatures 350K, 450K, and 550K, for 2.8 ns. These simulations were then analyzed using the same procedure: performing block averaging alone as well as double averaging over the velocity distributions, producing similar plots of $\overline{\Lambda}^{\chi}_{Phiv}$ vs τ and $\overline{\overline{\Lambda}}^{\chi}_{\Phi}$ vs τ for their respective structures for both full and tracer approaches. These plots were then fit to a $1/\tau$ functional form using linear regression, allowing for the estimate of σT values by taking the limit of the $1/\tau$ fit. From these, we similarly were able to estimate the full and tracer conductivity and generate Arrhenius plots. The Arrhenius plots were fit with a simple linear regression

of the form $1000/T$, which was used to estimate the activation energy. Due to that fact that these models are not comparable to the CS2 - CS6 models or one another, we cannot perform further triple averaging over Allegro models in this section.

In Figs. 22 and 23, we see the full and tracer $\overline{\Lambda}^{\chi}_{\Phi v}$ and $\overline{\overline{\Lambda}}^{\chi}_{\Phi}$ results for Allegro model SS1 with structure S1. In Fig.s 24 and 25, we see the full and tracer $\overline{\Lambda}^{\chi}_{\Phi v}$ and $\overline{\overline{\Lambda}}^{\chi}_{\Phi}$ results for Allegro model SS2 with structure S2.

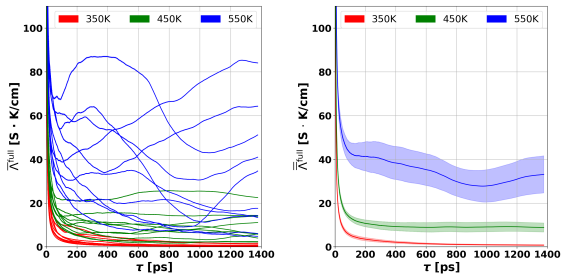


FIG. 22. Left plot shows $\overline{\Lambda}^{\text{full}}_{\Phi v}(\tau)$ versus time interval τ for structure S1 for Φ corresponding to Allegro model SS1 evaluated at at target temperatures 350K, 450K, and 550K each for 10 different initial velocity distributions. The standard error $\Delta\overline{\Lambda}^{\text{full}}_{\Phi v}(\tau)$ is indicated with light colored shading and is very small. Right plot shows the corresponding results for $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ with a bold line and the standard error $\Delta\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ with light colored shading at target temperatures 350K, 450K, and 550K.

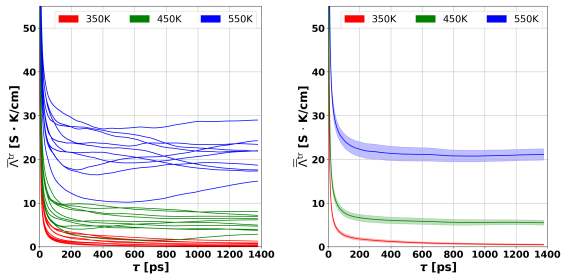


FIG. 23. Plots analgous to Fig. 22 for tracer interval functions.

In Figs. 26 and 27, we see examples of fits to $\overline{\overline{\Lambda}}(\tau)^{\chi}_{\Phi}$ for Allegro models SS1 and SS2 for both the full and tracer data for structures S1 and S2 respectively.

Here we see that the overall behavior found when analyzing the models with combined training still

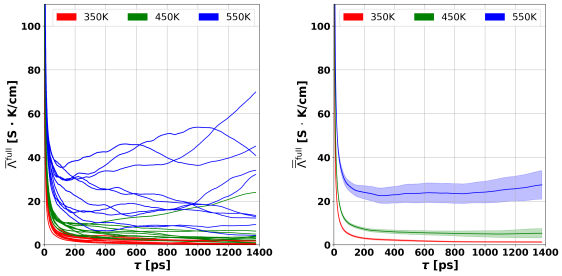


FIG. 24. Plots analgous to Fig. 22 with Φ corresponding to Allegro model SS2 evalued for reference structure S2.

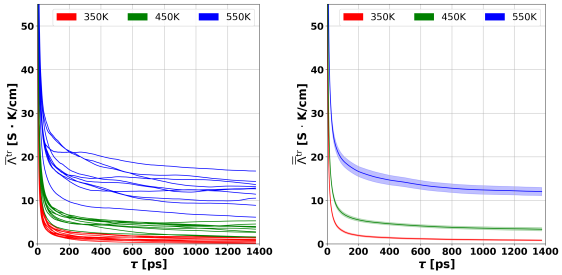


FIG. 25. Plots analgous to Fig. 24 for tracer interval functions.

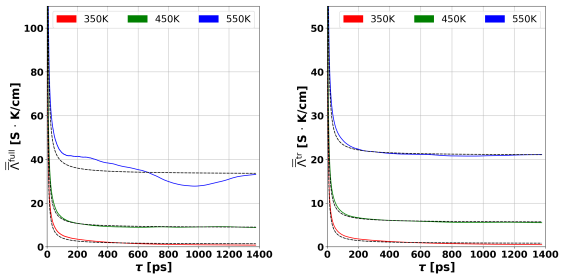


FIG. 26. Plots of the analytic fitting function defined in Eq. (33) indicated by the dashed lines for target temperatures 350K, 450K, and 550K. On the left, the fit and the function correspond to the doubly averaged interval function $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ with Φ represented Allegro model SS1. On the right, the fit and the function correspond to the analogous functions representing the tracer approximation. The evaluations were based on reference structure S1.

holds; namely that the full conductivity is still not well behaved when only averaging over block intervals, but is much better controlled when additionally averaging over multiple initial velocity distributions. We also see that the lower temperatures are much

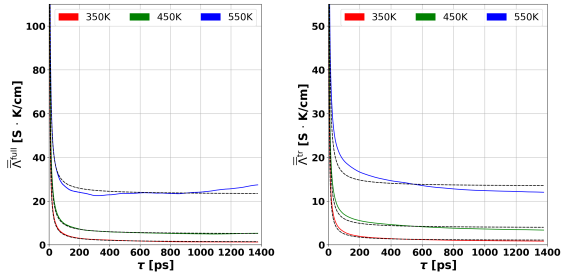


FIG. 27. Plots of the analytic fitting function defined in Eq. (33) indicated by the dashed lines for target temperatures 350K, 450K, and 550K. On the left, the fit and the function correspond to the doubly averaged interval function $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ with Φ represented Allegro model SS2. On the right, the fit and the function correspond to the analogous functions representing the tracer approximation. The evaluations were based on reference structure S2.

better controlled in general than the higher 550K temperature. The doubly averaged $\overline{\overline{\Lambda}}^{\chi}_{\Phi}$ vs. τ plots show again that there are still some issues involving the asymptotic behavior and fit precision, primarily at 550K. For these single structure models, we do not have multiple comparable models to average over, so we instead analyze the single models individually. In Fig. 28, we see the Arrhenius plots containing both full and tracer results for SS1 and SS2.

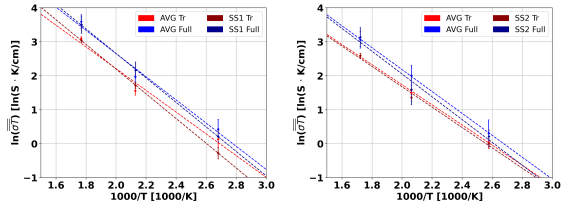


FIG. 28. Allegro model SS1 (left) and SS2 (right) average Arrhenius plots of full and tracer conductivity results for structures S1 and S2 respectively. Combined CS2 - CS6 model average results are included for comparison. Tracer conductivity estimates are given in red and full conductivity estimates are given in blue.

From these results, we can analyze the estimated conductivity for structure S1 and S2, as seen in Table XI, as well as the activation energies, as seen in Table X.

From these conductivity estimates, we see that these models again demonstrate the fact that the materials have appreciable ionic correlations based on the estimated Haven ratios being less than 1.0. For the full conductivity estimates, we have 3.33 mS/cm at 373.17K for structure S1 and 3.20 mS/cm at 387.91K for structure S2. Compared to the model averages of roughly 4.13 ± 1.18 mS/cm for structure S1 and 3.52 ± 1.38 mS/cm for structure S2 from the previous section using the models with combined S1 and S2 training, we see that the single models are in good agreement with the combined models since they are generally within the error of the combined model average. We also see similar errors for these models compared to the combined S1 and S2 trained models. For structure S1, we have errors of 0.56 - 15.00 mS/cm for the full conductivity. For structure S2, we have errors of 0.62 - 11.28 mS/cm for the full conductivity. Similarly, the activation energies from the single models are 310.56 meV for structure S1 and 289.02 for structure S2, which are within the error of the model average values from the combined training of 293.97 ± 41.32 and 277.58 ± 13.53. The single model activation errors are between 10.16 - 46.78 meV, which are similar to the errors present in the combined model training. These results suggest that the choice to combine structures S1 and S2 for training to improve the dataset diversity does not impact the predictive capacity when applying the model to a single structure compared to training on that structure alone.

TABLE X. Table containing the activation energy calculations for structures S1 and S2 based on the previous full and tracer conductivity estimations from Allegro models SS1 and SS2. The results from the triple averaged conductivity estimates of the CS2 - CS6 Allegro models are included for comparison, and are denoted by "AVG". Energy units are given in meV.

Model	Struct	E_a^{tr}	E_a^{full}
SS1	S1	313.28 ± 4.28	310.56 ± 10.16
AVG	S1	275.65 ± 40.51	293.97 ± 41.32
SS2	S2	257.89 ± 24.25	289.02 ± 46.78
AVG	S2	254.94 ± 10.05	277.58 ± 13.53

VII. CONCLUSIONS

In the present work, we used the Allegro software package [14, 15], based on a neural network architecture with localized atom pair representations,

TABLE XI. Table detailing the $\overline{\overline{\sigma}}$ and H_R results for the SS1 and SS2 small models investigated from LAMMPS molecular dynamics utilizing starting structures S1 and S2 respectively at target temperatures of 350K, 450K, and 550K. The combined CS2-CS6 model average is included for comparison, which is denoted by "AVG". Here temperatures are given in K, conductivity results are given in mS/cm, and H_R results are unitless.

Model	Struct	$T^{\rm targ}$	$\overline{\overline{T}}_\Phi$	$\overline{\overline{\sigma}}^{\rm tr}_\Phi$	$\overline{\overline{\sigma}}^{\rm full}_\Phi$	H_R
SS1	S1	350K	373.17 ± 0.31	2.04 ± 0.34	3.33 ± 0.56	0.61 ± 0.15
SS1	S1	450K	469.63 ± 0.25	11.92 ± 1.14	18.68 ± 4.39	0.64 ± 0.16
SS1	S1	550K	565.27 ± 0.13	36.96 ± 2.28	58.81 ± 15.00	0.63 ± 0.16
AVG	S1	350K	373.15 ± 0.50	3.06 ± 0.74	4.13 ± 1.18	0.74 ± 0.28
AVG	S1	450K	470.01 ± 0.30	10.11 ± 1.28	15.33 ± 3.89	0.66 ± 0.19
AVG	S1	550K	565.38 ± 0.30	39.24 ± 2.57	64.25 ± 13.45	0.61 ± 0.13
SS2	S2	350K	387.91 ± 0.34	2.58 ± 0.36	3.20 ± 0.62	0.81 ± 0.19
SS2	S2	450K	485.04 ± 0.09	8.00 ± 0.79	10.12 ± 4.41	0.79 ± 0.35
SS2	S2	550K	580.60 ± 0.08	22.98 ± 1.62	39.78 ± 11.28	0.58 ± 0.17
AVG	S2	350K	388.03 ± 0.45	2.71 ± 0.47	3.52 ± 1.38	0.77 ± 0.33
AVG	S2	450K	485.01 ± 0.20	9.27 ± 1.05	15.43 ± 4.58	0.60 ± 0.19
AVG	S2	550K	580.75 ± 0.22	22.96 ± 1.42	36.80 ± 9.00	0.62 ± 0.16

trained on trajectory configuration data generated with accurate density functional calculations on a sample ionic material system, to produce Allegro model interaction potentials. The Allegro model interaction potentials were used with the LAMMPS [16] software package and our own codes to evaluate various approximations to the full and tracer ionic conductivity within the Green-Kubo formalism [1–4]. The sample ionic material system in the study was based on two idealized crystalline electrolytes derived from the work of Deng and co-workers [23]. Taking into account differences in physical and numerical approximations between the present work and that of Deng and co-workers [23], we find rough agreement in estimates of ionic conductivity and in finding the Haven ratio to indicate correlated ionic motion for this system. The calculations involve many possible parameter choices involved with these computations including choices in machine learning hyperparameters, in molecular dynamics protocols, etc. Ideally, a materials focussed study would attempt to optimize these choices for the particular system of interest which was only partially explored in the present work. Instead, the focus has been on observing expected and unexpected results based on particular choices of parameter settings.

In order to achieve a level of reproducibility and to separate different components of Green-Kubo evaluation of ionic conductivity, the present work used a very conservative protocol for molecular dynamics simulation within the LAMMPS package. As described in Sec. V A, this included the use of the NVE ensemble and using CPU rather than GPU hardware in order to attain a level of reproducibility in the results. This allowed for the observation that the Lyapunov instability is reproducible on identical CPU hardware but differs on CPU hardware from different manufacturers (in our case AMD and Intel). For molecular dynamics runs on GPU hardware, the form of the Lyapunov instability differs on each run. In addition to maintaining consistent CPU hardware to ensure reproducibility in this manner, one also needs to maintain consistent software environments, including OS kernels, math libraries, etc. In this work, AMD CPU hardware was used to produce the block averaged $\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)$ and $\overline{\Lambda}^{\rm tr}_{\Phi v}(\tau)$ plots for 10 different initial velocity distributions in Figs. 7, 8, 9, 10, 22, 23, 24, and 25. One observation based on the present work, is that performing long computer simulations is not always the only challenge for understanding macroscopic phenomena. For example, in Secs. V B and VI C we noted that in long molecular dynamics simulations it is possible that Lyapunov instabilities can disrupt causal relationships in the computed particle trajectories and violate the premise of the Green-Kubo formula, Eq. (9) for evaluating ionic conductivity as shown in the single (block) average plots of $\overline{\Lambda}^{\rm full}_{\Phi v}(\tau)$. Interestingly, the temperature and total energy profiles of the runs shown in Figs. 3 and 4 for a particular initial velocity distribution shows no evidence of the Lyapunov instabilities across any of the temperatures tested. This behavior, is consistent with the literature [34] observation the Lyapunov instability is due to the nonlinearities in the equations of motion and cannot be eliminated by improving the accuracy of the nu-

merical time integration. The Lyapunov instability is clearly demonstrated for simulations at high target temperature (in this case 550 K) but is less evident for simulations at lower target temperatures (in this case 450 K and 350 K). For this reason and following recommendations of previous researchers [10, 11], we see that one important advantage of the efficiency of evaluating Allegro model potentials is that it makes it possible to perform molecular dynamics simulations at relatively low temperatures. This not only allows the simulations to more closely follow the experimental conditions, but also to drastically reduce the magnitude of the effects of Lyapunov instabilities present in the block averaged $\overline{\Lambda}^{\text{full}}_{\Phi v}(\tau)$ plots. While we noted, in a brief literature review, that several authors such as Posch and Hoover [66] and Skeel [67] have stressed the important impact of Lyapunov instabilities on molecular dynamics simulations, it is apparent that the effect is highly temperature dependent.

Another question explored in this work is to assess how well the Allegro model interaction potentials can represent the first principles results. This is discussed in Sec. VI B which shows that, because of the stochastic nature of the training algorithms, there is considerable variation of the Allegro error metrics. However Fig. 6 does show that we can find an observable trend between lower Allegro error metrics and the ability of the Allegro model to find the correct reference structures.

For most of the study, we have included two distinct but similar reference structures within the training set and this seems to work well. A simple test of Allegro models trained on single structures S1 or S2 yield conductivity and activation energy estimates that are roughly within those estimates from Allegro models trained on the combined S1 + S2 structures. Furthermore, we see that the Allegro error metrics are similar and the Λ plots exhibit similar behavior. One clear point regarding the training data is that it is important to include energies, forces, and strains in the training process in order to construct useful potential models.

On the other hand, the Allegro error metrics are not the only quantities that are needed to assess a given Allegro model interaction potential. In Sec. V C we showed one example of an Allegro model potential which, during a molecular dynamics simulation initiated at a relatively high target temperature (900 K) using LAMMPS , generated unphysical trajectories of the ions after ($\approx$ 1 ns) of simulation time. While, as mentioned above, there is good reason to perform Green-Kubo analyses with lower temperature simulations, the high temperature simulation can serve as a check on the Allegro model. From this experience, we recommend that before running systematic molecular dynamics simulations with a particular machine learned model potential, it is wise to perform high temperature screening runs. There is credible evidence that the reason for this type of failure comes from the molecular dynamics algorithm generating ion configurations that were not well represented in the training dataset. In observing the form of the basis functions used by Allegro shown in Fig. 2, we see that at small interparticle separations all the basis functions have a peak value. If the training set does not have sufficient examples of small interparticle separations in which the physical interparticle potential should be repulsive, the learned potential may not properly represent that repulsion. As mentioned in Sec. IV C, we find that by augmenting the training datasets with a small number of systematically designed configurations with small interparticle separations, we can reduce the number Allegro models that fail the 900K screening by exhibiting unphysical behavior. We also emphasize the importance of constructing the training data with care, using stringent convergence criteria in the first principles calculations, particularly in evaluation of the energies, forces, and stresses of each configuration of the dataset.

The conservative computational protocol used in this work is useful to demonstrate the Lyapunov instability effects, relative to the variability in machine learned potential models as well as variability due to initial velocity distributions in the NVE ensemble. However, it is clear that Lyapunov instability effects are part of the more general story. We fully expect that Green-Kubo analyses based on evaluating $\overline{\Lambda}^{\text{full}}_{\Phi v}(\tau)$, $\overline{\overline{\Lambda}}^{\text{full}}_{\Phi}(\tau)$ and $\overline{\overline{\Lambda}}^{\text{full}}(\tau)$ can be extended to other statistical ensembles such as NVT (constant number of particles, volume, and temperature) and NPT (constant number of particles, pressure, and temperature).

This study could be extended to explore other factors such as the effects of system size, the effects of different hyperparameter choices in constructing the machine learned models, and the effects of molecular dynamics ensemble representations. For example, modifying the simulation time of molecular dynamics runs and/or the number of initial velocity distributions could be considered for further study. Clearly, there are many opportunities offered by the the machine learning developments in general and by the Allegro package in particular.

ACKNOWLEDGMENTS

DCL and NAWH were supported by NSF Grant No. DMR-2242959 and by the Wake Forest University (WFU) High Performance Computing Facility, a centrally managed computational resource available to WFU researchers including faculty, staff, students, and collaborators with special thanks Dr. Sean Anderson for installing and updating software packages on the WFU-HPC computer system. CWT and BK were supported by Robert Bosch LLC and National Science Foundation, Office of Advanced Cyberinfrastructure (OAC), under Award No. 2118201.

Many people have helped us develop strategies for evaluating the Green-Kubo conductivies, including Dr. Angela Harper from OTI Luminonics, Professor Federico Grasselli from Unimore International, Professor Fred Salsbury from Wake Forest University, and Dr. Ryosuke Jinnouchi from Toyota Central R & D Labs. Professor Fan Yang from Wake Forest University helped us develop strategies for training and testing machine learning software. We would like also thank Dr. Normand Modine from Sandia National Laboratory, and Professors Alejandro Cárdenas-Avendaño and Greg Cook from Wake Forest University for discussions on Lyapunov instabilities. Additionally we would like to thank the MATSCI Community Discourse (MatSci.org) for their help with LAMMPS syntax.

[1] R. Kubo, Journal of the Physical Society of Japan **12**, 570 (1957).
[2] R. Kubo, M. Yokota, and S. Nakajima, Journal of the Physical Society of Japan **12**, 1203 (1957).
[3] M. S. Green, The Journal of Chemical Physics **20**, 1281 (1952).
[4] M. S. Green, The Journal of Chemical Physics **22**, 398 (1954).
[5] A. Marcolongo and N. Marzari, Physical Review Materials **1**, 025402 (2017).
[6] F. Grasselli and S. Baroni, Nature Physics **15**, 967 (2019).
[7] F. Grasselli and S. Baroni, The European Physical Journal B **94**, 160 (2021).
[8] F. Grasselli, The Journal of Chemical Physics **156**, 134705 (2022).
[9] L. Gigli, D. Tisi, F. Grasselli, and M. Ceriotti, Chemistry of Materials **36**, 1482 (2024).
[10] A. F. Harper, T. Huss, S. S. Köcher, and C. Scheurer, Faraday Discussions **255**, 411 (2025).
[11] S. Minami, A. Kutana, R. Jinnouchi, and R. Asahi, Physical Review Materials **9**, 103802 (2025).
[12] N. Kamaya, K. Homma, Y. Yamakawa, M. Hirayama, R. Kanno, M. Yonemura, T. Kamiyama, Y. Kato, S. Hama, K. Kawamoto, and A. Mitsui, Nature Materials **10**, 682 (2011).
[13] L. Kahle, A. Marcolongo, and N. Marzari, Energy & Environmental Science **13**, 928 (2020).
[14] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, Nature Communications **14**, 579 (2023).
[15] C. W. Tan, M. L. Descoteaux, M. Kotak, G. d. M. Nascimento, S. R. Kavanagh, L. Zichi, M. Wang, A. Saluja, Y. R. Hu, T. Smidt, A. Johansson, W. C. Witt, B. Kozinsky, and A. Musaelian, High-performance training and inference for deep equivariant interatomic potentials (2025), arXiv:2504.16068 [physics].
[16] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. In 'T Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, Computer Physics Communications **271**, 108171 (2022), URL for software: `https://www.lammps.org/`.
[17] Information Systems and Wake Forest University (WFU) High Performance Computing Facility (2021), URL: `https://doi.org/10.57682/g13z-2362`.
[18] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. De Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, Journal of Physics: Condensed Matter **21**, 395502 (2009).
[19] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. Dal Corso, S. De Gironcoli, P. Delugas, R. A. DiStasio, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. Otero-de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, Journal of Physics: Condensed Matter **29**, 465901 (2017), URL for software: `https://www.quantum-espresso.org`.
[20] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, Nature Communications **13**, 2453 (2022).
[21] A. Hjorth Larsen, J. Jørgen Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. Bjerre Jensen, J. Kermode, J. R. Kitchin, E. Leonhard Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. Bergmann Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, Journal of Physics: Condensed Matter **29**, 273002 (2017).
[22] Y.-W. Hu, I. D. Raistrick, and R. A. Huggins, Journal of the Electrochemical society **124**, 1240 (1977).
[23] Y. Deng, C. Eames, J.-N. Chotard, F. Lalère, V. Seznec, S. Emge, O. Pecher, C. P. Grey, C. Masquelier, and M. S. Islam, Journal of the American Chemical Society **137**, 9136 (2015).
[24] J. P. Perdew, A. Ruzsinszky, G. I. Csonka, O. A. Vydrov, G. E. Scuseria, L. A. Constantin, X. Zhou, and K. Burke, Phys. Rev. Lett. **100**, 136406 (2008), DOI:10.1103/PhysRevLett.100.136406.
[25] P. E. Blöchl, Phys. Rev. B **50**, 17953 (1994), DOI:https://doi.org/10.1103/PhysRevB.50.17953.
[26] N. A. W. Holzwarth, A. R. Tackett, and G. E. Matthews, Computer Physics Communications **135**, 329 (2001), DOI:https://doi.org/10.1016/S0010-4655(00)00244-7; Available from the website: http://pwpaw.wfu.edu.
[27] Supplemental Materials for this paper are available.
[28] P. Hohenberg and W. Kohn, Physical Review **136**,

B864 (1964), DOI:10.1103/PhysRev.136.B864.
[29] W. Kohn and L. J. Sham, Physical Review **140**, A1133 (1965), DOI:10.1103/PhysRev.140.A1133.
[30] M. Born and K. Huang, *Dynamical Theory of Crystal Lattices* (Oxford University Press, 1954) Appendix VIII expands the ideas presented by M. Born and R. Oppenheimer, *Ann. d. Phys.* **54**, 457 (1927).
[31] V. U. Nazarov and E. K. U. Gross, Journal of Chemical Theory and Computation , acs.jctc.5c02174 (2026).
[32] J. M. Haile, *Molecular Dynamics Simulation* (John Wiley Inc., 1992).
[33] M. E. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation*, 2nd ed. (Oxford University Press, 2023).
[34] D. Frenkel and B. Smit, *Understanding Molecular Simulation*, 3rd ed. (Academic Press, 2023).
[35] L. Onsager, Physical Review **37**, 405 (1931).
[36] L. Onsager, Physical Review **38**, 2265 (1931).
[37] K. Campaan and Y. Haven, Transactions of the Faraday Society **54**, 1498 (1958).
[38] Y. Haven and B. Verkerk, Physics and chemistry of glasses **6**, 38 (1965).
[39] A. D. Le Claire, in *Physical Chemistry An Advanced Treatise*, Vol. X (Academic Press, Inc., New York, 1970) pp. 261–330.
[40] G. E. Murch, Solid State Ionics **7**, 177 (1982).
[41] X. He, Y. Zhu, A. Epstein, and Y. Mo, npj Computational Materials **4**, 18 (2018).
[42] M. Otero-Lema, R. Lois-Cuns, M. A. Boado, H. Montes-Campos, T. Méndez-Morales, and L. M. Varela, Journal of Chemical Information and Modeling **65**, 3477 (2025).
[43] Note that, as mentioned by Frenkel and Smit in Chapter. 5.3.3, in some applications using block averaging, a requirement of statistical independence of sampling data involves more complicated equations than considered in this work.
[44] Philip R. Bevington and D. K. Robinson, *Data reduction and error analysis for the physical sciences*, 3rd ed. (McGraw Hill Higher Education , 2003).
[45] N. R. Mott and R. W. Gurney, *Electronic Processes in Ionic Crystals* (Dover Publications, Inc., 1964).
[46] J. Behler and M. Parrinello, Physical Review Letters **98**, 146401 (2007).
[47] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, Phys. Rev. Lett. **104**, 136403 (2010).
[48] J. Behler, Chemical Reviews **121**, 10037 (2021).
[49] R. Jacobs, D. Morgan, S. Attarian, J. Meng, C. Shen, Z. Wu, C. Y. Xie, J. H. Yang, N. Artrith, B. Blaiszik, G. Ceder, K. Choudhary, G. Csanyi, E. D. Cubuk, B. Deng, R. Drautz, X. Fu, J. Godwin, V. Honavar, O. Isayev, A. Johansson, B. Kozinsky, S. Martiniani, S. P. Ong, I. Poltavsky, K. Schmidt, S. Takamoto, A. P. Thompson, J. Westermayr, and B. M. Wood, Current Opinion in Solid State and Materials Science **35**, 101214 (2025).
[50] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).
[51] K. Hornik, Neural Networks **4**, 251 (1991).
[52] B. Kozinsky, A. Musaelian, A. Johansson, and S. Batzner, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (ACM, Denver CO USA, 2023) pp. 1–12.
[53] M. Geiger and T. Smidt, arXiv preprint arXiv:2207.09453 (2022).
[54] Nvidia, Nvidia/cuequivariance: Cuequivariance is a math library that is a collective of low-level primitives and tensor ops to accelerate widely-used models, like diffdock, mace, allegro and nequip, based on equivariant neural networks. also includes kernels for accelerated structure prediction.
[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019) pp. 8024–8035.
[56] Y. Xu and R. Goodacre, Journal of Analysis and Testing **2**, 249 (2018).
[57] E. Lopez, J. Etxebarria-Elezgarai, J. M. Amigo, and A. Seifert, Analytica Chimica Acta **1275**, 341532 (2023).
[58] Y. Park, J. Kim, S. Hwang, and S. Han, Journal of chemical theory and computation **20**, 4857 (2024).
[59] J. Gasteiger, J. Gross, and S. Günnemann, Directional Message Passing for Molecular Graphs (2022), arXiv:2003.03123 [physics, stat].
[60] Z. Yan, Z. Fan, and Y. Zhu, Journal of Chemical Information and Modeling **66**, 1406 (2026).
[61] J. L. Lebowitz, J. K. Percus, and L. Verlet, Physical Review **153**, 250 (1967).
[62] L. Verlet, Physical Review **159**, 98 (1967).
[63] L. Verlet, Physical Review **165**, 201 (1968).
[64] D. Levesque and L. Verlet, Journal of Statistical Physics **72**, 519 (1993).
[65] A. M. Liapunov, *Stability of Motion* (Academic Press, 1966) translation by Flavian Abramovici and Michael Shimshoni from the Department of Applied Mathematics, Weizmann Institute of Science, Rehovoth, Israel, of 1892 manuscript by A. M. Liapunov who founded modern stability theory.
[66] H. A. Posch and W. G. Hoover, Physical Review A **39**, 2175 (1989).
[67] R. D. Skeel, SIAM Journal on Scientific Computing **31**, 1363 (2009).
[68] L. D. S. Oliveira and P. A. Greaney, Physical Review E **95**, 023308 (2017).
[69] N. Molinari, Y. Xie, I. Leifer, A. Marcolongo, M. Kornbluth, and B. Kozinsky, Physical Review Letters **127**, 025901 (2021).
[70] X. Liu, X. Guo, and Q. Liao, Polymer **290**, 126519 (2024).
[71] G. Toraman, D. Fauconnier, and T. Verstraelen, Journal of Chemical Information and Modeling **65**, 10445 (2025).

[72] P. Pegolo, E. Drigo, F. Grasselli, and S. Baroni, The Journal of Chemical Physics **162**, 064111 (2025).

[73] R. Krechetnikov and J. E. Marsden, Reviews of Modern Physics **79**, 519 (2007).

Bibliography

[1] A. O'Donovan, A. Grant, N. Soulopoulos, R. Fisher, J. Lyu, S. Mi, S.-H. Wong, M. Yang, H. Zhou, S. Wachche, S. Kalaichelvan, C. Lubis, L. Hong, M. Brolly, S.-H. Wong, E. Stoikou, A. Leach, J. Shi, Y. Sekine, K. Ampofo, P. Xu, K. Kareer, L. Quong, V. Nunes, R. Rabioglio, S. Mothe, T. Kawahara, T. Noble, A. Vagneur-Jones, M. Hales, N. Wang, A. Logan, A. Izadi-Najafabadi, and C. McKerracher, *Electric Vehicle Outlook 2026*, Technical report, BloombergNEF (2026).

[2] J. Fleischmann, M. Hanicke, E. Horetsky, D. Ibrahim, S. Jautelat, M. Linder, P. Schaufuss, L. Torscht, and A. van de Rijt, *Battery 2030: Resilient, sustainable, and circular*, Technical report, McKinsey and Company (2023).

[3] P. Albertus, V. Anandan, C. Ban, N. Balsara, I. Belharouak, J. Buettner-Garrett, Z. Chen, C. Daniel, M. Doeff, N. J. Dudney, B. Dunn, S. J. Harris, S. Herle, E. Herbert, S. Kalnaus, J. A. Libera, D. Lu, S. Martin, B. D. McCloskey, M. T. McDowell, Y. S. Meng, J. Nanda, J. Sakamoto, E. C. Self, S. Tepavcevic, E. Wachsman, C. Wang, A. S. Westover, J. Xiao, and T. Yersak, *Challenges for and pathways toward Li-metal-based all-solid-state batteries*, ACS Energy Letters **6**, 1399 (2021), DOI: 10.1021/acsenergylett.1c00445.

[4] P. K. Singh, S. Mallick, G. A. Kaur, S. Balayan, and A. Tiwari, *John b. good-enough's pioneering contributions towards advancements in photo-rechargeable lithium batteries*, Nano Energy **128**, 109792 (2024), DOI: https://doi.org/10.1016/j.nanoen.2024.109792.

[5] J. Xie and Y.-C. Lu, *A retrospective on lithium-ion batteries*, Nature Communications **11**, 2499 (2020), DOI: 10.1038/s41467-020-16259-9.

[6] C. Costa, E. Lizundia, and S. Lanceros-Méndez, *Polymers for advanced lithium-ion batteries: State of the art and future needs on polymers for the different battery components*, Progress in Energy and Combustion Science **79**, 100846 (2020), DOI: https://doi.org/10.1016/j.pecs.2020.100846.

[7] M. Pigłowska, B. Kurc, M. Galiński, P. Fuć, M. Kamińska, N. Szymlet, and P. Daszkiewicz, *Challenges for safe electrolytes applied in lithium-ion cells—a review*, Materials **14** (2021), DOI: 10.3390/ma14226783.

[8] D. H. P. Kang, M. Chen, and O. A. Ogunseitan, *Potential environmental and human health impacts of rechargeable lithium batteries in electronic waste*, Environmental Science & Technology **47**, 5495 (2013), DOI: 10.1021/es400614y.

[9] X. Yu, W. Li, V. Gupta, H. Gao, D. Tran, S. Sarwar, and Z. Chen, *Current challenges in efficient lithium-ion batteries' recycling: A perspective*, Global Challenges **6**, 2200099 (2022), DOI: https://doi.org/10.1002/gch2.202200099.

[10] J. Zhang, L. Zhang, F. Sun, and Z. Wang, *An overview on thermal safety issues of lithium-ion batteries for electric vehicle application*, IEEE Access **6**, 23848 (2018), DOI: 10.1109/ACCESS.2018.2824838.

[11] A. Alkhalidi, M. K. Khawaja, and S. M. Ismail, *Solid-state batteries, their future in the energy storage and electric vehicles market*, Science Talks **11**, 100382 (2024), DOI: https://doi.org/10.1016/j.sctalk.2024.100382.

[12] A. Machín, C. Morant, and F. Márquez, *Advancements and challenges in solid-state battery technology: An in-depth review of solid electrolytes and anode innovations*, Batteries **10**, 29 (2024), DOI: 10.3390/batteries10010029.

[13] A. Joshi, D. K. Mishra, R. Singh, J. Zhang, and Y. Ding, *A comprehensive review of solid-state batteries*, Applied Energy **386**, 125546 (2025), DOI: https:

//doi.org/10.1016/j.apenergy.2025.125546.

[14] Y. Li and N. A. W. Holzwarth, *First-principles simulations to understand the structural and electrolyte properties of idealized $Li_{7.5}B_{10}S_{18}X_{1.5}$ (X=Cl, Br, I)*, Phys. Rev. Mater. **6**, 045403 (2022), DOI: 10.1103/PhysRevMaterials.6.045403.

[15] Y. Li and N. A. W. Holzwarth, *First-principles simulations of Li boracites $Li_4B_7O_{12}Cl$ and $Li_5B_7O_{12.5}Cl$*, Phys. Rev. Mater. **6**, 025401 (2022), DOI: 10.1103/PhysRevMaterials.6.025401.

[16] W. Jeitschko, T. A. Bither, and P. E. Bierstedt, *Crystal structure and ionic conductivity of Li boracites*, Acta Crystallographica Section B **33**, 2767 (1977), DOI: 10.1107/S0567740877009443.

[17] D. Tan, F. Wang, T. Pietsch, M. A. Grasser, T. Doert, and M. Ruck, *Low-Temperature Ionothermal Synthesis of Li-Ion Conductive $Li_4B_7O_{12}Cl$ Solid-State Electrolyte*, ACS Applied Energy Materials **2**, 5140 (2019), DOI: 10.1021/acsaem.9b00812.

[18] T. Katsumata, Y. Aoki, K. Fushimi, K. Otsuka, K. Ueda, and Y. Inaguma, *Synthesis and ionic conductivity of li boracites, $Li_4B_7O_{12}Cl$ and $Li_4B_4Al_3O_{12}Cl_{1-x}Br_x$*, Solid State Ionics **380**, 115921 (2022), DOI: https://doi.org/10.1016/j.ssi.2022.115921.

[19] K. Kajihara, N. Tezuka, M. Shoji, J. Wakasugi, H. Munakata, and K. Kanamura, *$Li_4B_4M_3O_{12}Cl$ (M = Al, Ga): An electrochemically stable, lithium-ion-conducting cubic boracite with substituted boron sites*, Bulletin of the Chemical Society of Japan **90**, 1279 (2017), DOI: 10.1246/bcsj.20170242.

[20] K. Kaup, K. Bishop, A. Assoud, J. Liu, and L. F. Nazar, *Fast ion-conducting thioboracite with a perovskite topology and argyrodite-like lithium substructure*,

Journal of the American Chemical Society **143**, 6952 (2021), DOI: 10.1021/jacs.1c00941, pMID: 33929830.

[21] C. Keffer, A. D. Mighell, F. Mauer, H. E. Swanson, and S. Block, *Crystal structure of twinned low-temperature lithium phosphate*, Inorganic Chemistry **6**, 119 (1967), DOI: 10.1021/ic50047a027.

[22] Y.-W. Hu, I. Raistrick, and R. Huggins, *Ionic conductivity of lithium phosphate-doped lithium orthosilicate*, Materials Research Bulletin **11**, 1227 (1976), DOI: https://doi.org/10.1016/0025-5408(76)90025-8.

[23] Y. Deng, C. Eames, J.-N. Chotard, F. Lalère, V. Seznec, S. Emge, O. Pecher, C. P. Grey, C. Masquelier, and M. S. Islam, *Structural and mechanistic insights into fast lithium-ion conduction in Li_4SiO_4–Li_3PO_4 solid electrolytes*, Journal of the American Chemical Society **137**, 9136 (2015), DOI: 10.1021/jacs.5b04444, pMID: 26118319.

[24] H.-J. Deiseroth, S.-T. Kong, H. Eckert, J. Vannahme, C. Reiner, T. Zaiß, and M. Schlosser, *Li_6PS_5X: A class of crystalline Li-rich solids with an unusually high Li+ mobility*, Angewandte Chemie International Edition **47**, 755 (2008), DOI: https://doi.org/10.1002/anie.200703900.

[25] H.-J. Deiseroth, J. Maier, K. Weichert, V. Nickel, S.-T. Kong, and C. Reiner, *Li_7PS_6 and Li_6PS_5X (X: Cl, Br, I): Possible three-dimensional diffusion pathways for lithium ions and temperature dependence of the ionic conductivity by impedance measurements*, Zeitschrift für anorganische und allgemeine Chemie **637**, 1287 (2011), DOI: https://doi.org/10.1002/zaac.201100158.

[26] C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong, *Graph networks as a universal machine learning framework for molecules and crystals*, Chemistry of Materials

31, 3564 (2019), DOI: 10.1021/acs.chemmater.9b01294.

[27] M. Born and R. Oppenheimer, *Zur quantentheorie der molekeln*, Annalen der Physik **389**, 457 (1927), DOI: https://doi.org/10.1002/andp.19273892002.

[28] D. Tong, *11.1.3 The Born-Oppenheimer Approximation*, pp. 365–367, Cambridge University Press (2025).

[29] M. Born and K. Huang, *Dynamical theory of Crystal Lattices, by Max Born and Kun Huang*, Clarendon Press (1968).

[30] P. Hohenberg and W. Kohn, *Inhomogeneous electron gas*, Phys. Rev. **136**, B864 (1964), DOI: 10.1103/PhysRev.136.B864.

[31] W. Kohn and L. J. Sham, *Self-consistent equations including exchange and correlation effects*, Phys. Rev. **140**, A1133 (1965), DOI: 10.1103/PhysRev.140.A1133.

[32] R. M. Martin, *Part II: Density Functional Theory*, pp. 129–214, Cambridge University Press (2020).

[33] J. P. Perdew and Y. Wang, *Accurate and simple analytic representation of the electron-gas correlation energy*, PRB **45**, 13244 (1992).

[34] J. P. Perdew, K. Burke, and M. Ernzerhof, *Generalized gradient approximation made simple*, PRL **77**, 3865 (1996), erratum – Phys. Rev. Let. **78**, 1396 (1997).

[35] J. P. Perdew, A. Ruzsinszky, G. I. Csonka, O. A. Vydrov, G. E. Scuseria, L. A. Constantin, X. Zhou, and K. Burke, *Restoring the density-gradient expansion for exchange in solids and surfaces*, Phys. Rev. Lett. **100**, 136406 (2008), DOI: 10.1103/PhysRevLett.100.136406.

[36] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna,

I. Carnimeo, A. Dal Corso, S. de Gironcoli, P. Delugas, R. A. DiStasio, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. Otero-de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, *Advanced capabilities for materials modelling with QUANTUM ESPRESSO*, Journal of Physics: Condensed Matter **29**, 465901 (2017), DOI: 10.1088/1361-648X/aa8f79.

[37] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, *QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials*, Journal of Physics: Condensed Matter **21**, 395502 (2009), DOI: 10.1088/0953-8984/21/39/395502.

[38] P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, S. de Gironcoli, P. Delugas, F. Ferrari Ruffino, A. Ferretti, N. Marzari, I. Timrov, A. Urru, and S. Baroni, *QUANTUM ESPRESSO toward the exascale*, The Journal of Chemical Physics **152**, 154105 (2020), DOI: 10.1063/5.0005082.

[39] F. Bloch, *Über die quantenmechanik der elektronen in kristallgittern*, Zeitschrift für Physik **52**, 555 (1929), DOI: 10.1007/BF01339455.

[40] N. W. Ashcroft and N. D. Mermin, *Solid State Physics*, Cengage (2021).

[41] W. Massa and R. O. Gould, *Crystal structure determination*, Springer (2011).

[42] *Materials data on NaCl by materials project* (2020), DOI: 10.17188/1199028.

[43] H. Hellmann, *A new approximation method in the problem of many electrons*, The Journal of Chemical Physics **3**, 61 (1935), DOI: 10.1063/1.1749559.

[44] J. C. Phillips and L. Kleinman, *New method for calculating wave functions in crystals and molecules*, Phys. Rev. **116**, 287 (1959), DOI: 10.1103/PhysRev.116.287.

[45] D. Vanderbilt, *Soft self-consistent pseudopotentials in a generalized eigenvalue formalism*, Phys. Rev. B **41**, 7892(R) (1990), DOI: 10.1103/PhysRevB.41.7892.

[46] P. E. Blöchl, *Projector augmented-wave method*, Phys. Rev. B **50**, 17953 (1994), DOI: 10.1103/PhysRevB.50.17953.

[47] N. Holzwarth, A. Tackett, and G. Matthews, *A projector augmented wave (paw) code for electronic structure calculations, part I: atompaw for generating atom-centered functions*, Computer Physics Communications **135**, 329 (2001), DOI: https://doi.org/10.1016/S0010-4655(00)00244-7.

[48] A. Togo and I. Tanaka, *First principles phonon calculations in materials science*, Scripta Materialia **108**, 1 (2015), DOI: https://doi.org/10.1016/j.scriptamat.2015.07.021.

[49] A. Togo, *First-principles phonon calculations with phonopy and phono3py*, Journal of the Physical Society of Japan **92**, 012001 (2022), DOI: 10.7566/JPSJ.92.012001.

[50] A. Togo, L. Chaput, T. Tadano, and I. Tanaka, *Implementation strategies in phonopy and phono3py*, Journal of Physics: Condensed Matter **35**, 353001

(2023), DOI: 10.1088/1361-648X/acd831.

[51] S. Nosé, *A unified formulation of the constant temperature molecular dynamics methods*, The Journal of Chemical Physics **81**, 511 (1984), DOI: 10.1063/1.447334.

[52] W. G. Hoover, *Canonical dynamics: Equilibrium phase-space distributions*, Phys. Rev. A **31**, 1695 (1985), DOI: 10.1103/PhysRevA.31.1695.

[53] D. Evans and B. Holian, *The Nosé–Hoover thermostat*, The Journal of Chemical Physics **83**, 4069 (1985), DOI: 10.1063/1.449071.

[54] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, *Molecular dynamics with coupling to an external bath*, The Journal of Chemical Physics **81**, 3684 (1984), DOI: 10.1063/1.448118.

[55] D. Frenkel and B. J. Smit, *Understanding molecular simulation from algorithms to applications*, Acad. Press (2013).

[56] M. E. Tuckerman, *Statistical mechanics: Theory and molecular simulation*, Oxford University Press (2023).

[57] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, *Lammps - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales*, Comp. Phys. Comm. **271**, 108 (2022), DOI: 10.1016/j.cpc.2021.108171.

[58] G. E. P. Box and M. E. Muller, *A note on the generation of random normal deviates*, Annals of Mathematical Statistics **29**, 610 (1958).

[59] M. S. Green, *Markoff random processes and the statistical mechanics of time-dependent phenomena*, The Journal of Chemical Physics **20**, 1281 (1952), DOI: 10.1063/1.1700722.

[60] R. Kubo, *Statistical-mechanical theory of irreversible processes. I. general theory and simple applications to magnetic and conduction problems*, Journal of the Physical Society of Japan **12**, 570 (1957), DOI: 10.1143/JPSJ.12.570.

[61] R. Kubo, M. Yokota, and S. Nakajima, *Statistical-mechanical theory of irreversible processes. II. response to thermal disturbance*, Journal of the Physical Society of Japan **12**, 1203 (1957), DOI: 10.1143/JPSJ.12.1203.

[62] M. S. Green, *Markoff random processes and the statistical mechanics of time-dependent phenomena. II. irreversible processes in fluids*, The Journal of Chemical Physics **22**, 398 (1954), DOI: 10.1063/1.1740082.

[63] F. Grasselli and S. Baroni, *Topological quantization and gauge invariance of charge transport in liquid insulators*, Nature Physics **15**, 967 (2019), DOI: 10.1038/s41567-019-0562-0.

[64] F. Grasselli and S. Baroni, *Invariance principles in the theory and computation of transport coefficients*, The European Physical Journal B **94**, 160 (2021), DOI: 10.1140/epjb/s10051-021-00152-5.

[65] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, Journal of Big Data **8**, 53 (2021), DOI: 10.1186/s40537-021-00444-8.

[66] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Networks **2**, 359 (1989), DOI: https://doi.org/

10.1016/0893-6080(89)90020-8.

[67] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, ArXiv **abs/1511.08458** (2015).

[68] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, *Relational inductive biases, deep learning, and graph networks* (2018).

[69] D. Bacciu, F. Errica, A. Micheli, and M. Podda, *A gentle introduction to deep learning for graphs*, Neural Networks **129**, 203 (2020), DOI: 10.1016/j.neunet.2020.06.006.

[70] D. Hendrycks and K. Gimpel, *Gaussian error linear units (GELUs)* (2023), https://arxiv.org/abs/1606.08415.

[71] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, *A review on weight initialization strategies for neural networks*, Artificial Intelligence Review **55**, 291 (2022), DOI: 10.1007/s10462-021-10033-z.

[72] J. Buzas, *An analysis of loss functions for binary classification and regression* (2023), https://arxiv.org/abs/2301.07638.

[73] R. Caruana, S. Lawrence, and C. Giles, *Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping*, in T. Leen, T. Dietterich, and V. Tresp, eds., *Advances in Neural Information Processing Systems*, volume 13, MIT Press (2000).

[74] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization* (2017), DOI: https://doi.org/10.48550/arXiv.1412.6980.

[75] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization* (2019), DOI: https://doi.org/10.48550/arXiv.1711.05101.

[76] P. Kidger and T. Lyons, *Universal approximation with deep narrow networks*, in J. Abernethy and S. Agarwal, eds., *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2306–2327, PMLR (2020).

[77] J. Behler and M. Parrinello, *Generalized neural-network representation of high-dimensional potential-energy surfaces*, Phys. Rev. Lett. **98**, 146401 (2007), DOI: 10.1103/PhysRevLett.98.146401.

[78] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, *E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials*, Nature Communications **13**, 2453 (2022), DOI: 10.1038/s41467-022-29939-5.

[79] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, *Learning local equivariant representations for large-scale atomistic dynamics*, Nature Communications **14**, 579 (2023), DOI: 10.1038/s41467-023-36329-y.

[80] M. Tinkham, *Group theory and Quantum Mechanics*, Dover Publications (2003).

[81] D. S. Dummit and R. M. Foote, *Abstract algebra*, John Wiley & Sons, Inc (2004).

[82] J. D. H. Smith, *Introduction to abstract algebra*, Chapman & Hall CRC (2024).

[83] J. J. Rotman, *An introduction to the theory of groups Joseph J. Rotman*, Springer (1995).

[84] C. W. Tan, M. L. Descoteaux, M. Kotak, G. de Miranda Nascimento, S. R. Kavanagh, L. Zichi, M. Wang, A. Saluja, Y. R. Hu, T. Smidt, A. Johansson, W. C. Witt, B. Kozinsky, and A. Musaelian, *High-performance training and inference for deep equivariant interatomic potentials*, Digital Discovery **5**, 1558 (2026), DOI: 10.1039/d5dd00423c.

[85] J. Gasteiger, J. Groß, and S. Günnemann, *Directional message passing for molecular graphs* (2022).

[86] H. Wang, X. Guo, L. Zhang, H. Wang, and J. Xue, *Deep learning inter-atomic potential model for accurate irradiation damage simulations*, Applied Physics Letters **114** (2019), DOI: 10.1063/1.5098061.

[87] Z. D. Hood, A. U. Mane, A. Sundar, S. Tepavcevic, P. Zapol, U. D. Eze, S. P. Adhikari, E. Lee, G. E. Sterbinsky, J. W. Elam, and J. G. Connell, *Multifunctional coatings on sulfide-based solid electrolyte powders with enhanced processability, stability, and performance for solid-state batteries*, Advanced Materials **35**, 2300673 (2023), DOI: https://doi.org/10.1002/adma.202300673.

[88] G.-J. Jang, R. Rajagopal, S. Kang, and K.-S. Ryu, *Preparation of argyrodite* $Li_{6-2x}Zn_xPS_{5-x}O_xCl$ *with improved electrochemical performance and air stability for all-solid-state batteries*, Journal of Alloys and Compounds **957**, 170273 (2023), DOI: https://doi.org/10.1016/j.jallcom.2023.170273.

[89] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and

S. Chintala, *Pytorch: An imperative style, high-performance deep learning library* (2019).

[90] J. D. Hunter, *Matplotlib: A 2d graphics environment*, Computing in Science & Engineering **9**, 90 (2007), DOI: 10.1109/MCSE.2007.55.

Curriculum Vitae

D. Cory Lynch

+1-336-684-5536 | lyncdw19@wfu.edu

Graham, NC, 27253

Education

- Ph.D. Physics, Wake Forest University, Winston-Salem, NC 2026
- B.S. Physics, University of North Carolina Greensboro, Greensboro, NC 2017
- A.S., Alamance Community College, Graham, NC 2012

Experience

- Q.A. Lab Technician, Berkshire, Whitsett, NC 2017-2019

Publications

- D. Cory Lynch, Chuin Wei Tan, Boris Kozinsky, and N. A. W. Holzwarth, *A study of the use of machine learning methodology for computing ionic conductivity in an example solid state electrolyte material*, (2026), preprint.
- D. Cory Lynch, Yan Li, Pieremanuele Canepa, and N. A. W. Holzwarth, *Computational investigation of the structural and electrolyte properties of the extended family of lithium (thio)boracite materials: $Li_4B_7O_{12}Cl$ and beyond*, Phys. Rev. Materials, (2024), DOI: 10.1103/PhysRevMaterials.8.065401, online.

Presentations

- *Enabling the Investigation of the Conductivity of Select Lithium (Thio)Boracite Materials Using the Allegro Machine Learned Interatomic Potential Package*, APS Meeting, (March 2026).
- *A Study of Advancements in Ionic Conductivity Methods and Machine Learned Interatomic Potentials to Calculate Ionic Conductivities with a Model Solid Electrolyte $(Li_4SiO_4)_x(Li_4PO_4)_{1-x}$*, APS Meeting, (March 2025).
- *Simulations and Analyses of Li_4 and Li_6 Boracites and Thioboracites as Promising Li Ion Conducting Electrolytes*, 242nd ECS Meeting, (October 2022).

Skills

- Software experience: Quantum Espresso, Phonopy, VESTA, LAMMPS, Allegro
- Solid electrolyte material investigation including structural optimization, phonon analysis, molecular dynamics, and training machine learned interatomic potentials
- Programming experience: Python, C++, Mathematica