UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Predicting Properties of Crystals

Author:
Kostiantyn LAPCHEVSKYI

Supervisor:
Dr. Tess Eleonora SMIDT

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Predicting Properties of Crystals

by Kostiantyn LAPCHEVSKYI

Abstract

Crystalline structures are vital to the modern technology. Yet, we are still only starting to figure out how to properly estimate their directional properties using machine learning techniques. In order to improve that, I build upon the theory and codebase of Euclidean Neural Networks (networks equivariant to 3D rotations). The main contributions of this work are: a derivation of the decompositon/reconstruction equations of elastic tensor that enables using it as a train target, optimized CUDA implementation of the core operation `PeriodicConvolution` that makes it fast and scalable, and an analysis of the trends of geometric structures and electronic properties of the crystal in Materials Project Database and how these trends impact hyperparameters for convolutional neural network architectures such as Euclidean Neural Networks.

Supplementary materials can be found at: `https://github.com/L-sky/Master_Thesis`

Acknowledgements

I am thankful to my collaborators Mario Geiger and Ben Miller for all their patience and time they spent on me. And even more so I am grateful to my scientific advisor Tess Smidt and my mom Olena, only because of your continuous support through it all, this manuscript sees the light of day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Many wonders of the modern world harness the emergent phenomena that arise in condensed matter. To see this you do not need to look any further than your smartphone. The touchscreen is a liquid crystal. Beneath it, the integrated circuit and image sensor are made from single crystal wafers of semiconductors. The battery is made of complex layers of polycrystalline and polymer materials.

The properties of known materials set the physical limits on what technology can achieve, and finding new ones that allow to push the limits is a nontrivial task. Recently celebrated with the Nobel Prize in Chemistry 2019, lithium-ion batteries required more than a decade (1970's - 1980's) worth of improvements, particularly of the material of cathode, before achieving commercial viability [1].

With the exponential increase in computing power, computational physics and chemistry methods, such as density functional theory [2], have become widespread and are readily used for practical materials design. Material properties that once could only be obtained through experiments in laboratories, nowadays can be estimated reasonably accurate via computer modelling.

Yet, modeling many technologically critical phenomena (that rely on many-body quantum interactions) still remains out of practical reach. Accurate methods scale poorly with a number of atoms in a structure - cubic increase in required compute or worse [3]. Depending on the method, structure of interest, and requested accuracy, one calculation may take hours, days or even weeks on high-performance computing resources. The development of the new commercially viable materials has still become progressively more costly.

The pharmaceutical industry faces similar issues. It takes years to decades and billions US dollars to develop one eventually approved drug [4]. Recently, machine learning has been extensively applied in an attempt to alleviate bottlenecks in drug discovery pipelines [5, 6].

Most machine learning approaches for chemistry focus on a single molecule (often represented as topological graphs) with limited numbers of atoms and atom types. Although model considers a single molecule, values of the targeted property are usually averaged over the different conformations - distinct spatial arrangements of said molecule, particular conformations are hard to discern. In contrast, for the crystals, spatial arrangement is crucial, and relatively small changes can have dramatic effect - for example, graphite versus diamond. This means that building models that can efficiently and naturally process 3D geometry is of upmost importance for applying neural networks to the challenges of materials science.

1.2 Thesis structure

In this thesis, I build upon the theoretical basis of Euclidean Neural Networks (which naturally handle geometry and the datatypes of properties of physical systems) and `e3nn` software implements. In the remainder of this chapter I give background for the topics discussed in the subsequent chapters. In Chapter 2, I analyze Materials Project Database (de facto the only widespread benchmark dataset for crystal properties prediction). In particular, I focus on how it constraints the choice of representation / network hyperparameters. In Chapter 3, I overview theory of the Euclidean Neural Networks, how `e3nn` works and what are the limitations. Later in the chapter, I present my improvements to the implementation. In Chapter 4, I provide the decomposition of elastic tensor into the components that are compatible with `e3nn`, as well as reconstruction.

Note: Writing a thesis takes time. For that reason, Chapter 3 is based on the now legacy release called `se3nn`: `https://github.com/mariogeiger/se3cnn`

The contemporary release of `e3nn`: `https://github.com/e3nn/e3nn` retains the same theoretical basis, but some changes were made to the implementation.

1.3 Background

1.3.1 Crystals

A crystal structure is composed of a unit cell that periodically repeats (infinitely) in all directions given by defining vectors $\vec{A}, \vec{B}, \vec{C}$ (see Figure 1.2). A unit cell itself is an oblique prism that contains particular arrangement of atoms: 3D positions and atom types.

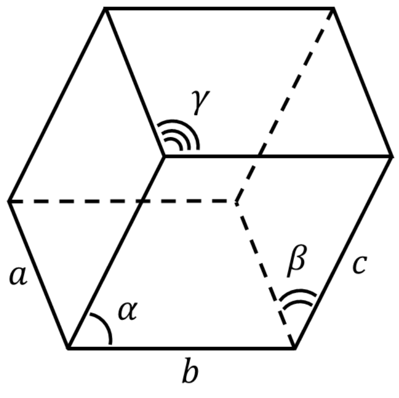

FIGURE 1.1: Unit cell given by side lengths and angles

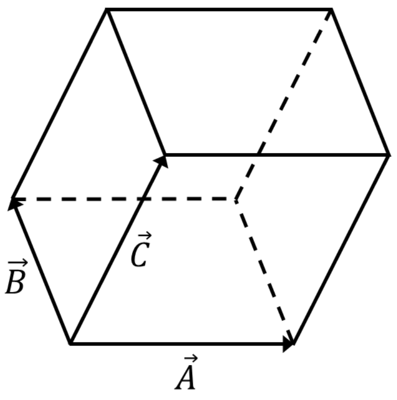

FIGURE 1.2: Unit cell given by vectors

A unit cell can be equivalently specified either with triplet of vectors (see Figure 1.2) or with side lengths and pairwise angles (see Figure 1.1). The later definition is useful for classification of structures into one of the seven lattice types (see Table 1.1). Lattice types can serve as a proxy to help determine the number of independent components in a higher order tensors (see for example number of independent components of the elastic tensor - column C_{ijkn} in Table 1.1).

TABLE 1.1: Lattice types [7]

Lattice type	Side lengths	Angles	C_{ijkn} [8]
Cubic	$a = b = c$	$\alpha = \beta = \gamma = 90°$	3
Tetragonal	$a = b \neq c$	$\alpha = \beta = \gamma = 90°$	6, 7
Orthorhombic	$a \neq b \neq c$	$\alpha = \beta = \gamma = 90°$	9
Hexagonal	$a = b$	$\alpha = \beta = 90°, \gamma = 120°$	5
Monoclinic	$a \neq c$	$\alpha = \gamma = 90°, \beta \neq 90°$	13
Rhombohedral	$a = b = c$	$\alpha = \beta = \gamma \neq 90°$	6, 7
Triclinic	otherwise		21

If inequalities for side lengths and angles in Table 1.1 to be forfeited, following hierarchical tree can be arranged (see Figure 1.3).

$$\begin{tikzcd}
 & Triclinic \arrow[drr] \arrow[dddl] & & & \\
 & & & Monoclinic \arrow[dl] \arrow[ddr] & \\
 & & Orthorhombic \arrow[d] & & \\
Rhombohedral \arrow[dr] & & Tetragonal \arrow[dl] & & Hexagonal \\
 & Cubic & & &
\end{tikzcd}$$

FIGURE 1.3: Hierarchy of lattice types from lowest symmetry to highest symmetry.

1.3.2 Equivariance

A function $f : \mathbb{R}^N \to \mathbb{R}^M$ is said to be **equivariant** to the group G if for any group action $g \in G$ and $x \in \mathbb{R}^N$:

$$f\left(D_N(g)x\right) = D_M(g)f(x) \tag{1.1}$$

This essentially means that order in which f and g applied can be interchanged. While g itself is an abstract element of a group, $D(g)$ is its matrix representation in the particular vector space. Group actions retain vector space, e.g. $D_N(g) : \mathbb{R}^N \to \mathbb{R}^N$.

The more commonly used property of **invariance** is a just a special case of Equation 1.1 when $D_M(g)$ is an identity.

1.3.3 Related work

The following are references on neural networks architectures relevant to the subject of this thesis, ordered by data of appearance. Brief descriptions are given next to each reference. (The release date of the first preprint used if available.)

MPNN [9] (2017) - "Neural Message Passing for Quantum Chemistry". Generalized and unified under the one framework prior approaches to graph convolutional neural networks.

SchNet [10] (2017) - "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions". Showed advantage of the continuous radial basis functions (as opposed to discrete). Calculated partial derivative of the output (energy) with respect to the input (coordinates) to get force, and used it as an additional target.

CGCNN [11] (2017) - "Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties". Explored different approaches on how to define neighborhood of an atom and thus form crystal graph. Middle ground between explicit feature engineering and deep learning approach.

Tensor field networks [12] (2018) - "Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds". Introduced formalism for networks (3D point clouds) that are equivariant to continuous 3D rotations.

Clebsch-Gordan Networks [13] (2018) - "Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network". Performs spherical convolutions in Fourier space using the formalism of irreducible representations of $SO(3)$.

3D Steerable CNNs [14] (2018) - "3D Steerable CNNs: Learning RotationallyEquivariant Features in Volumetric Data". Introduced formalism for networks (voxels) that are equivariant to continuous 3D rotations, and gating mechanism (see Figure 3.1).

MEGNet [15] (2018) - "Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals". Extended MPNN appoarch by introducing state variables that are shared among all nodes in the graph. As name suggests, state variables allow to express shared state (temperature, external magnetic field, etc.)

e3nn [16] (2018, ongoing) - "A modular framework for neural networks with Euclidean symmetry" (software). Originated as a combined implementation of approaches presented in "Tensor field networks" and "3D Steerable CNNs".

Cormorant [17] (2019) - "Cormorant: Covariant Molecular Neural Networks". Utilized alternative (as compared to [12, 14, 16]) basis - complex spherical harmonics - to formulate network equivariant to continuous 3D rotations. Used tensor products between features to express 2-body, and more generally many-body interactions.

SE(3) Equivariant Neural Networks for Regression on Molecular Properties: The QM9 Benchmark [18] (2020). Applied `e3nn` to the QM9 molecular dataset, and showed that equivariant features improve predictions on inherently vector quantities (dipole moment).

Chapter 2

Materials Project Database

The Materials Project Database [19, 20] (release **V2019.11**) comprises density functional theory (DFT) calculations of 124,515 crystalline structures and includes calculated properties such as formation energy, band gap, elasticity tensor, piezoelectric tensor, and others (check [21] for an exhaustive list). The Materials Project performs the DFT calculations with the Vienna Ab Initio Software Package (VASP) [22, 23, 24] using the generalized gradient approximation (GGA) functional of Perdew, Burke, and Ernzerhof (PBE) [25] and a Hubbard correction (+U) [26] for specific transition metals oxides, fluorides and sulfides. 39.2% of the structures are tied to the Inorganic Crystal Structure Database (ICSD) [27, 28] that contains both simulated and experimentally obtained data (95% of the tied structures are experimental).

In this section, I show how to choose convolutional neural network hyperparameters in a data-driven way, explore caveats of modelling properties and perform some general analysis on structure composition.

2.1 Bugs

Being an active and evolving endeavor, the Materials Project Database is not devoid of bugs. The ones that I personally uncovered:

- Some structures had attribute 'theoretical' set to 'None' whereas it should be 'True' (meaning that the structure has not been tied to an experimentally obtained one) [29]. (fixed)
- For 11 structures, U-values used for Hubbard correction are permuted [30]. (affected structures currently have attribute 'is_compatible' set to False, hence no longer being downloaded as a part of the database under the default settings)
- For 3158 structures, value of the attribute 'run_type' matches neither of the expected 'GGA' nor 'GGA+U' [31]. (expected to be fixed in the next release)

I exclude structures affected by the issue with invalid (displayed?) 'run_type' (3158 structures, 2.5% of the database) from the further analysis.

2.2 How to chose hyperparameters

Hyperparameters of the interest that I am going to focus on:

- number of layers in a network
- radial cutoff

The radial cutoff of a continuous convolutional layer defines the radius of the sphere around atom within which another atoms are considered to be its "neighbors" (see Figure 2.1 (left)). The underlying assumption implied by using convolutional layers is that "interactions are predominantly local" $\rightarrow$ "only neighbor atoms propagate information to each other". This assumption allows to improve from quadratic to linear scaling in compute and memory with respect to the linear increase in the number of atoms in a unit cell, which makes runs on larger structures practically feasible. Concept of the "neighborhood" as a volume within a sphere is widely used besides e3nn, for example see Refs. [10, 15].

Additional convolutional layers result in an increase of the effective "neighborhood", referred to as the receptive field, although it does not remain spherical (see Figure 2.1). Linear increase in the number of layers gives linear increase of the reception field "radius", and corresponds to the linear increase in memory and compute. In comparison, linear increase of radial cutoff corresponds to the cubic increase in memory and compute.

Under the limited resources, it appears favorable to decrease the radial cutoff and increase the number of layers. However, for neural network to be useful it is important to preserve the ability of the atoms to communicate information between each other. This imposes lower bound on the radial cutoff.

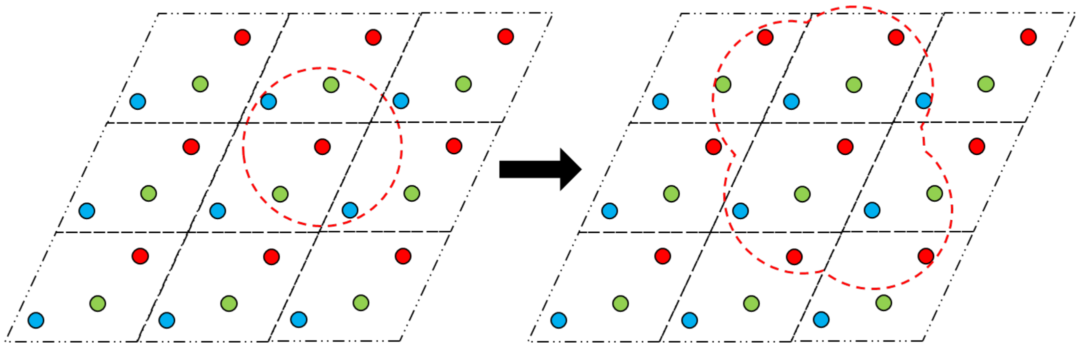

FIGURE 2.1: Expansion of the reception field after the first layer.

2.2.1 Radial cutoff

In this section, I am going to argue that 5 Å is a good choice for the radial cutoff.

The base premise is that atoms must be able to exchange information, otherwise the whole construction loses its validity - respective structures must be discarded. For some model implementations, like MEGNet [32], this fail-safe is hardwired - the model would throw an explicit error in presence of isolated atoms. However, the exclusion of isolated single atoms is not a sufficiently strict rule. Structures can have disconnected clusters; this case violates the premise that atoms must be able to exchange information, but in a more subtle way, each individual atom propagates information, but clusters do not.

Obviously, rejecting too many structures is undesirable. As can be seen from Figures 2.2 and 2.3, rejection rate remains fairly small.

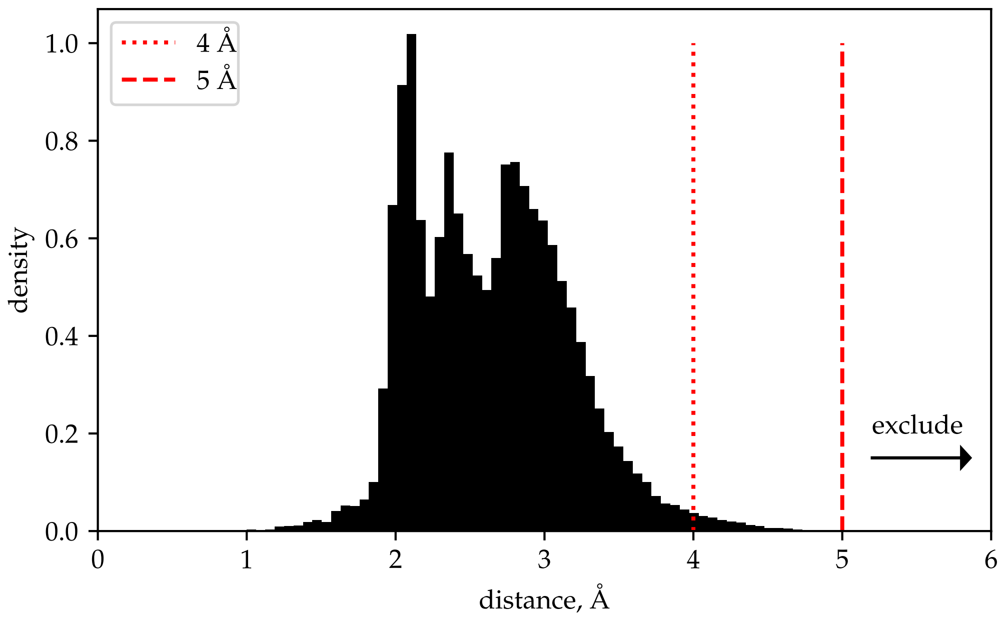

FIGURE 2.2: Distribution of distances atom - closest neighbor.
Clipped at 6 Å. For 174 structures distance is bigger. Maximum distance: 13.4 Å.
Atom is isolated if the distance exceeds the radial cutoff.

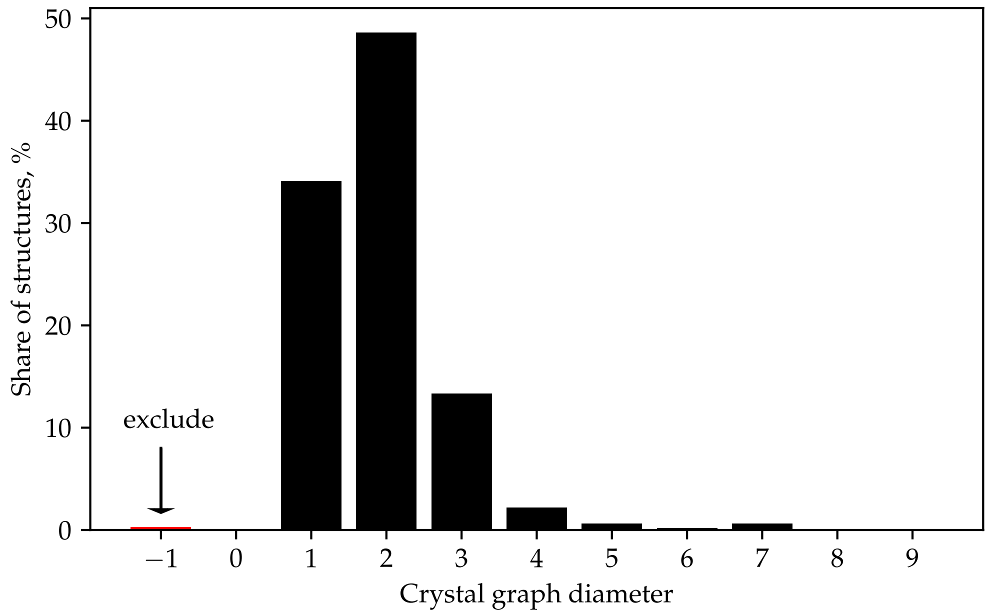

FIGURE 2.3: Distribution of diameters of crystal graphs for $r_{cut} = 5.0$ Å.
Clipped at 9. For 152 structures diameter is bigger. Maximum finite diameter is 32.
"-1" marks the case of disconnected components - infinite diameter.
Diameter is the max length among shortest paths between nodes in the graph.

At the time of writing, there is no merit in pushing radial cutoff further. Excluded structures have no particular significance, while difference in required compute resources would be considerable. Moreover, for larger radial cutoffs, it rapidly becomes more common to have multiple copies of the same atom (from translated unit cells) in the "neighborhood" (as evident from Figures 2.4, 2.5). The same features get used multiple times, resulting not only in a diminished returns on compute, but also in a heighten likeliness of a signal being unstable (see Appendix B.2.2).

2.2.2 Number of layers

Number of layers should be big enough for each atom in the unit cell to consolidate information from the whole unit cell. Diameter of the graph (see Figure 2.3) is the smallest value that satisfies the condition. For Materials Project Database, 4 layers with a radial cutoff of 5 Å is sufficient to cover 98% of structures.

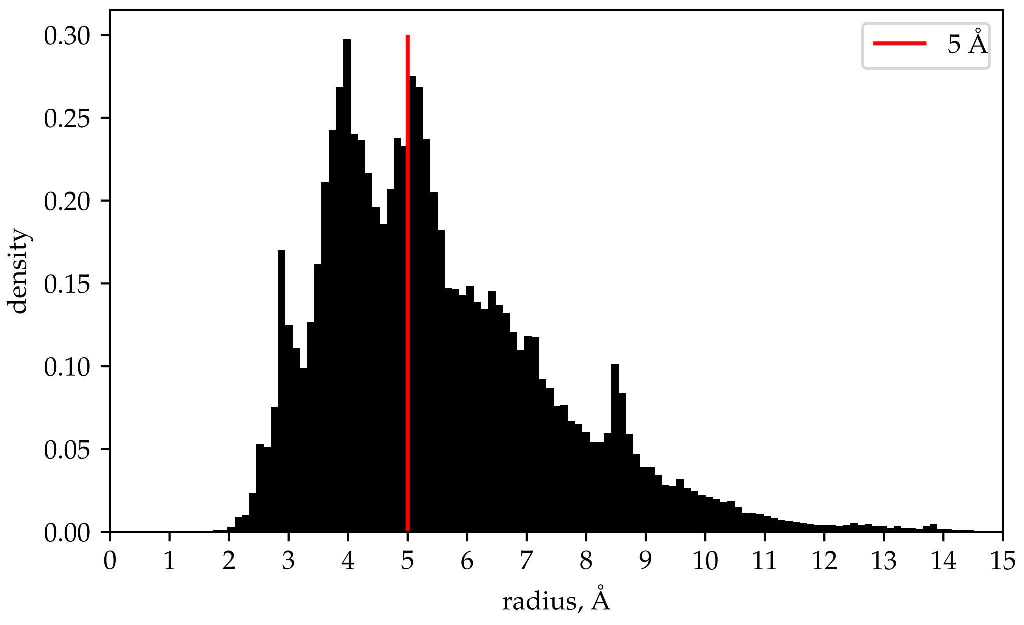

FIGURE 2.4: Distribution of radii of the largest sphere that fits in unit cell. Clipped at 15 Å. For 107 structures radius is bigger. Maximum: 24.7 Å. At most one copy of an atom appears in the "neighborhood" if radial cutoff is less than the radius.

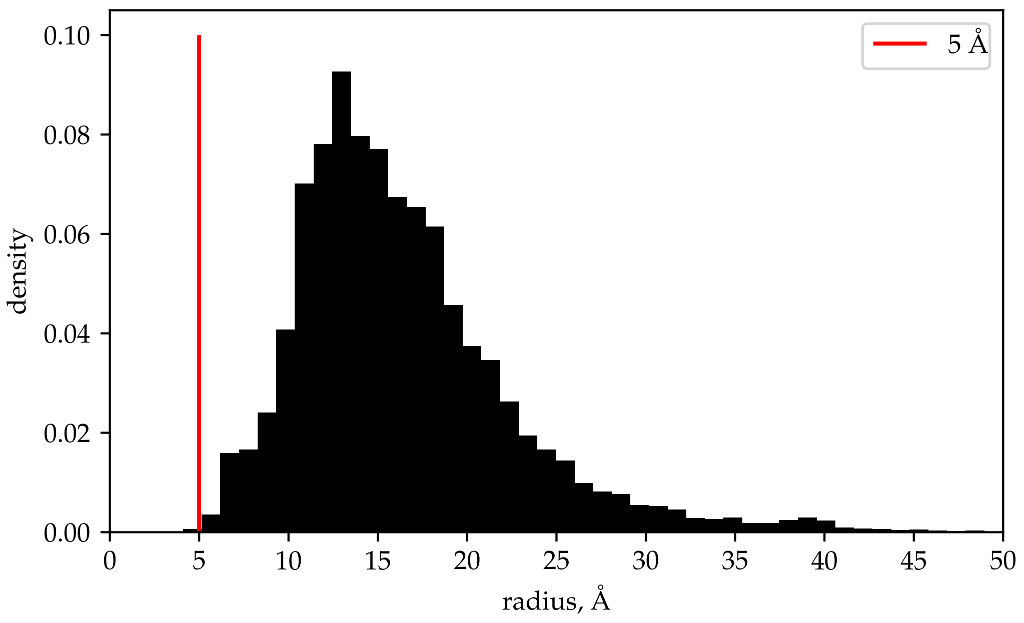

FIGURE 2.5: Distribution of radii of the smallest sphere enclosing unit cell. Clipped at 50 Å. For 718 structures radius is bigger. Maximum: 211.5 Å. Every atom in original unit cell appears in the "neighborhood" at least once if the radial cutoff exceeds the radius.

2.3 Properties

2.3.1 Band gap

The band gap is the minimum amount of energy required to excite an electron from valence band to conductive band [33]. Electrons in conductive band participate in conductivity. In simplified version, materials with zero band gap are conductors (usually metals) - pass electrical current; with "small" band gap (semi-conductors) - pass electrical current if suitable electric field applied to the material, with "big" band gap (insulators) - do not pass electrical current. There is no solid line between "small" and "big" band gap. For purposes of this work, I will only make distinction between zero and non-zero values. While the band gaps predicted by DFT at the level of GGA are know to be inaccurate due to well understood theoretical limitations, it is still useful to predict these values for coarse guidance in materials design.

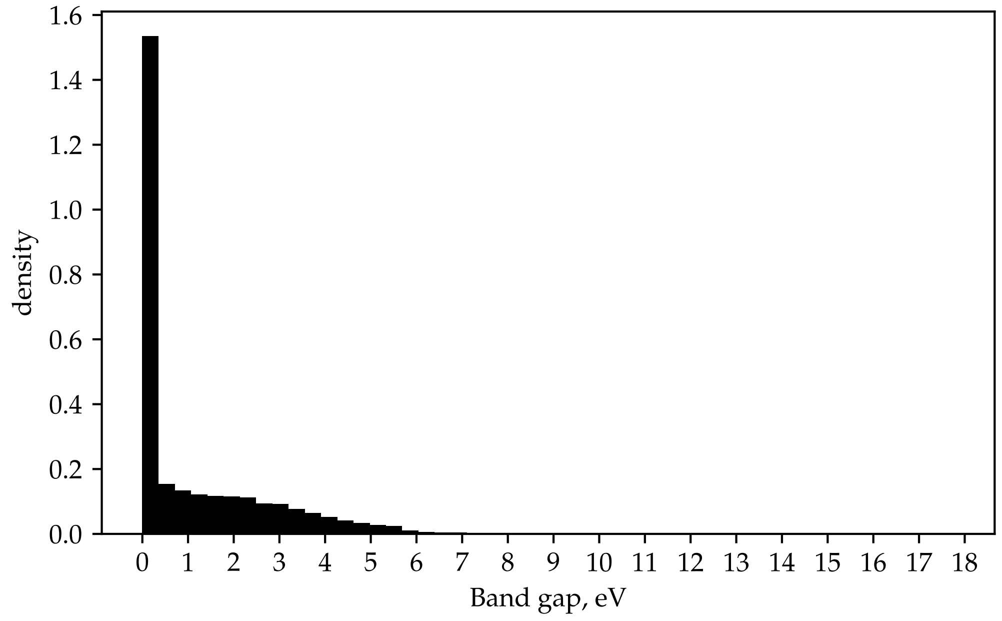

FIGURE 2.6: Distribution of band gap.

From Figure 2.6, it is apparent that band gap distribution is extremely skewed. 45% of materials have zero band gap, making it reasonable to model the distribution as a product of Bernoulli distribution and some other distribution. I claim that it is going to be advantageous for network training stability to have distribution modeled in the following form:

$$\text{Band gap} \sim B(p)\{\sigma(D(0,1)+\mu)\}^2 \tag{2.1}$$

where $B(p)$ - Bernoulli distribution; $D(0,1)$ - standardized distribution of square root of non-zero band gap (see Figure 2.7); μ and σ are respectively mean and standard deviation of the distribution of non-zero band gap.

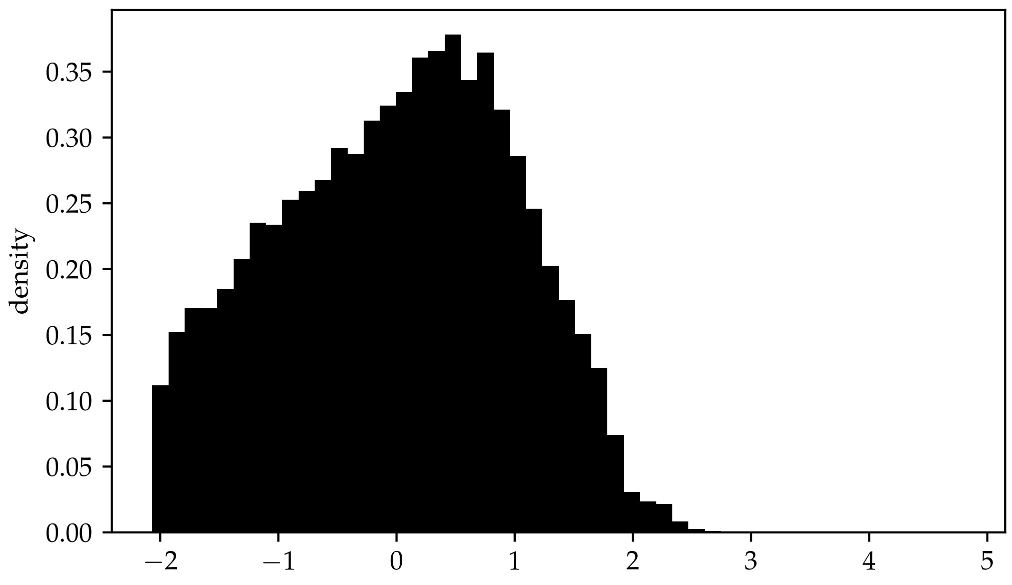

FIGURE 2.7: Distribution of standardized square root of non-zero band gap. mean = 1.274, standard deviation = 0.612 (before standardization), skew = −0.097, kurtosis = −0.802

2.3.2 Formation energy

Total formation energy is a difference in energies between structure as a whole and its constituent parts. Further into negatives the formation energy is, more stable the structure is. Conversely, positive values indicate that structure is prone to decay.

Note: Hereafter, the "formation energy" refers to the formation energy per atom.

Figure 2.8 is visually akin to bi-modal distribution and in fact, the peaks can be explained based on the run type used in the simulation (see Figure 2.9). Credit goes to Simon Verret for suggesting this reason.

Whether the difference between sub-distributions is due to the innate properties of materials for which either of run types has been chosen, or due to simulation method itself, which would imply systematic error - remains a mystery. It can be resolved via comparison to the calculations based on theoretical model of higher order. Unfortunately, said calculations are too computationally expensive for the scope this work. As for now, we shall refrain from exploiting bi-modality. The distribution can still be standardized in way as if it was uni-modal.

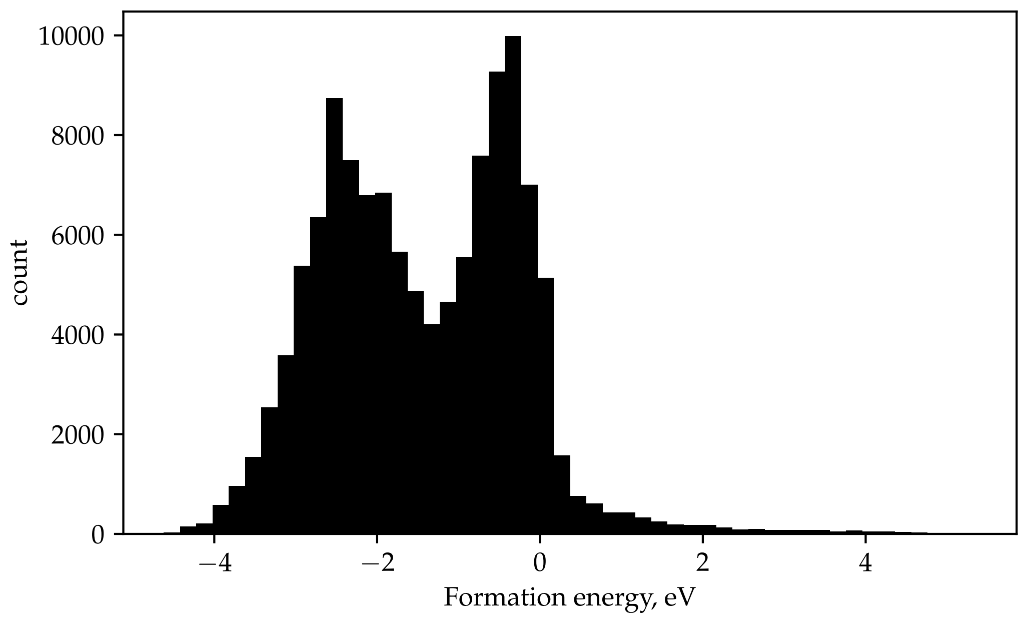

FIGURE 2.8: Distribution of formation energy.

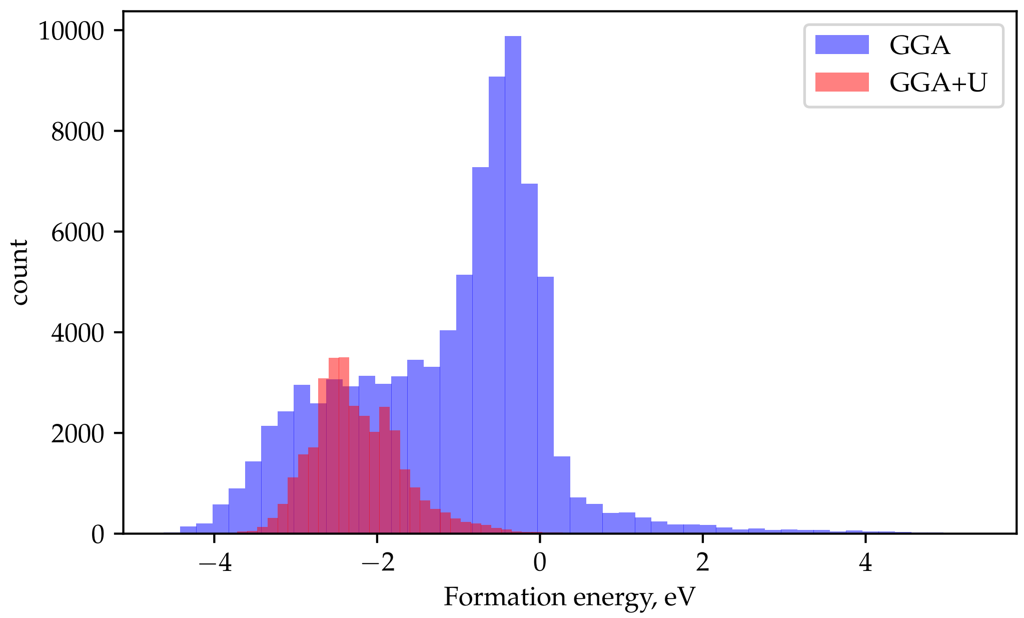

FIGURE 2.9: Distribution of formation energy by run type.

2.4 Extras

2.4.1 Structure sizes

The distribution of the number of atoms in a unit cell follows distinctive pattern due to parity (see Figure 2.10). Analogous behaviour has been observed earlier for ICSD [34]. In Ref. [34], author argues that parity pattern may be explained through symmetry broken centered Bravais lattices [7] (which is, in a sense, artifact of how unit cells are constructed), quote:

"... breaking some symmetries in a face-centered lattice may lead to the necessity to describe it by a supercell, corresponding to either a base-centered lattice (... factor of 2 in N_{at}) or a simple lattice (... factor of 4 in N_{at}). ... for symmetry-broken supercells of base- and body-centered lattices ... factor of 2, when the broken symmetry necessitates the description by a simple Bravais lattice" [34]. (N_{at} - number of atoms in a unit cell)

As for a decaying number of examples for more populated unit cells, it is expected as DFT calculations have cubic scaling in required compute with linear increase in N_{at} [3].

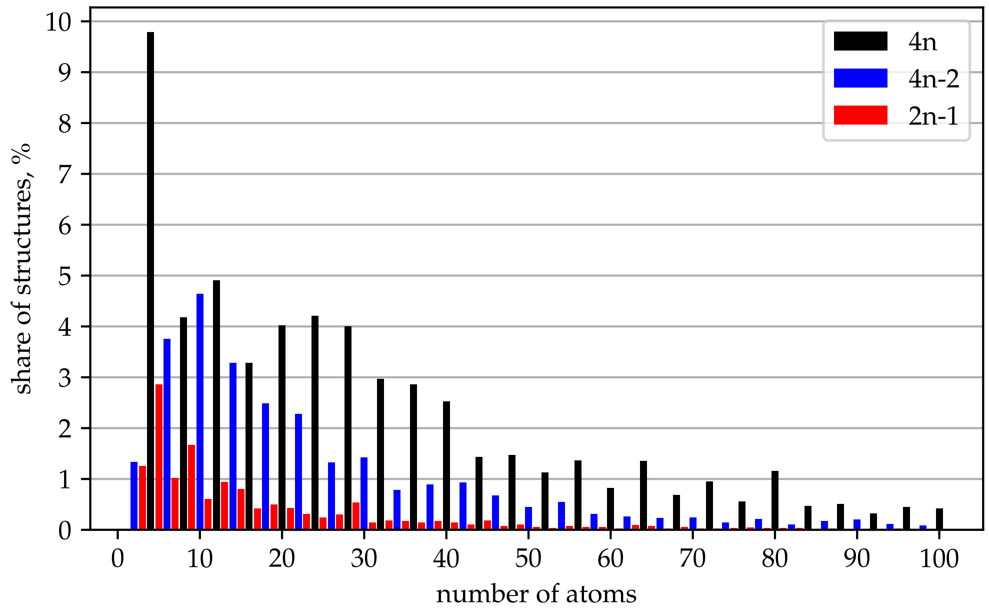

FIGURE 2.10: Distribution of the number of atoms in a unit cell. Clipped at 100. 3.1% of structures has more than 100 atoms in unit cell. Maximal number of atoms in a unit cell is 444. Parity is marked with a color.

2.4.2 Atomic composition

Presence of the different atom types in MP is not uniform (see Figure 2.11), with noble gases and radioactive elements being expectably underrepresented (Tables 2.1, 2.2). Noble gases are chemically inert, the only thing keeping a crystal comprised of noble gas elements together is Van der Waals force (which is relatively weak). Radioactive elements disrupt structure formation through decay.

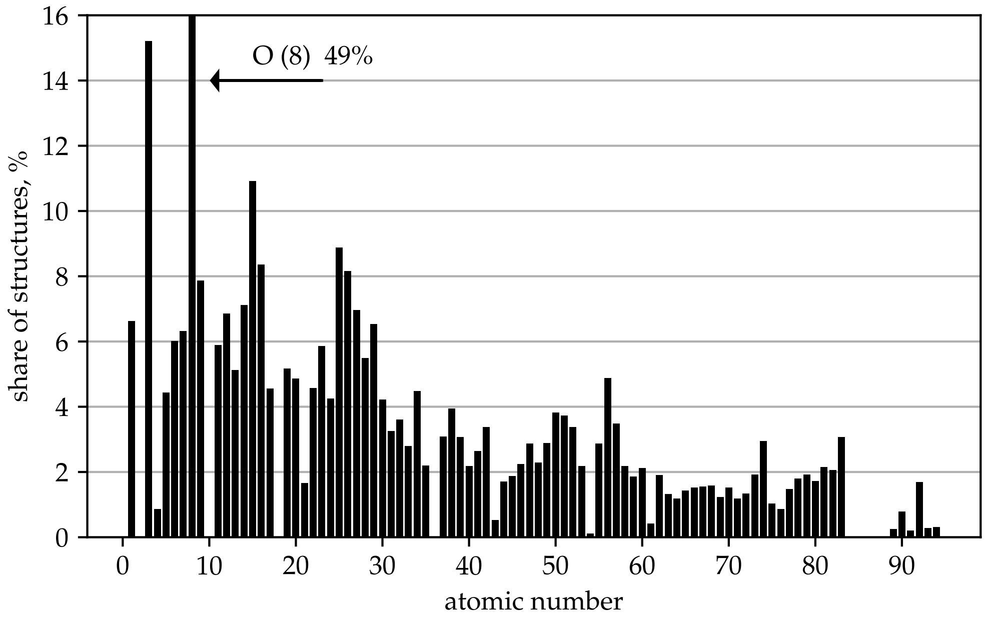

FIGURE 2.11: Presence of the atom types across structures.

TABLE 2.2: Radiactive elements

Element	Structures
Tc (43)	644
Pm (61)	513
Po (84)	0
At (85)	0
Rn (86)	0
Fr (87)	0
Ra (88)	0
Ac (89)	296
Th (90)	954
Pa (91)	250
U (92)	2038
Np (93)	338
Pu (94)	385

TABLE 2.1: Noble gases

Element	Structures
He (2)	5
Ne (10)	0
Ar (18)	1
Kr (36)	13
Xe (54)	143

Note: All radioactive elements have only unstable isotopes, however meaning of the "unstable" vary. *Tc* can have half-life over four million years, while *Fr* - only 22 minutes.

Chapter 3

Euclidean Neural Networks (E(3)NN)

Euclidean Neural Networks (E(3)NNs) are a general class of neural networks that are equivariant to the symmetries of 3D Euclidean space, 3D translation, rotations, and (optionally) inversion [12, 13, 14]. Equivariance means that order of action of neural network and these operations (translations, rotations, (optionally) inversion) on the input can be interchanged. This built-in equivariance guarantees that E(3)NNs can identify patterns in 3D in any orientation, avoiding the need for expensive (500 fold) data augmentation typically required for 3D convolutional neural networks to emulate rotation equivariance. Additionally, because these networks are equivariant at every layer, these networks can represent equivariant functions with substantially fewer parameters than traditional methods.

Crystal symmetries in 3D are characterized by the 230 space groups and tabulated in the International Tables of Crystallography [35]. These space groups are subgroups of 3D Euclidean symmetry $E(3)$. Thus, any symmetry of any crystal given as an input to E(3)NN, will be preserved, as desired.

The preexisting code base of E(3)NN [36] includes a `PeriodicConvolution` as well as a toy example on how to use it for crystal structures, but it is not optimized as needed for real-world cases. Medium-sized network (on GPU) triggers out-of-memory exception when given a single structure with more than 60 atoms in a unit cell, and having that many atoms is not rare (see Figure 2.10). Speed is also an issue - estimated time for one train epoch on full Materials Project dataset exceeds 12 hours. In order to alleviate it, I created optimized CUDA routine for `PeriodicConvolution` that is fast and scalable.

In this chapter, I briefly overview the theory behind E(3)NN, pinpoint aspects in the previous implementation that caused issues, and present my optimized solution.

3.1 Theory

Euclidean neural networks are a subset of convolutional neural networks where the filters are constrained to be separable into a radial function and angular function where the scalar radial function is learned and the angular functions are spherical harmonics, $K_{lm}(\vec{r}) = R_{(l)}(|r|)Y_{lm}(\hat{r})$. There are additional mathematical consequences due to this filter choice that effects how we combine filter and feature information as described in Appendix A.

3.1.1 Real Spherical Harmonics

Real spherical harmonics are equivariant to rotations in 3D:

$$Y_{\ell m}(\Omega + \delta\Omega) = \sum_{m'=-\ell}^{\ell} D^{\ell}_{mm'}(\mathfrak{R}(\delta\Omega))Y_{\ell m'}(\Omega) \equiv D^{\ell}_{mm'}(\mathfrak{R}(\delta\Omega))Y_{\ell m'}(\Omega) \tag{3.1}$$

where $\Omega \equiv (\theta, \phi)$ - polar and azimutal angles, $\delta\Omega$ - rotation angle, $Y_{\ell m}$ - real spherical harmonic, $D^{\ell}_{mm'}(\mathfrak{R}(\delta\Omega))$ - matrix representation of rotation $\mathfrak{R}(\delta\Omega)$ in the space of real spherical harmonics of rotational order ℓ (also referred as rotational matrix over real spherical harmonics [37, 38]).

Real spherical harmonics are defined as follows (arguments θ, φ implied):

$$Y_{\ell m} = \begin{cases} \sqrt{2}(-1)^m Im[Y_\ell^{|m|}] & \text{if } m < 0 \\ Y_\ell^0 & \text{if } m = 0 \\ \sqrt{2}(-1)^m Re[Y_\ell^m] & \text{if } m > 0 \end{cases} \tag{3.2}$$

where Y^{ℓ}_{m} - complex spherical harmonic.

In `e3nn`, real spherical harmonics are additionally multiplied by factor $(-1)^{-\ell}$:

$$Y_{\ell m} \to (-1)^{\ell} Y_{\ell m} \tag{3.3}$$

Complex spherical harmonics:

$$Y_\ell^m(\theta,\varphi) = (-1)^m \sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} P_\ell^m(\cos\theta) e^{im\varphi} \tag{3.4}$$

where $(-1)^m$ - QM normalization, P_ℓ^m - Associated Legendre Polynomials.

Associated Legendre Polynomials:

$$\begin{gathered} P_\ell^m(z) = (1-z^2)^{m/2}\frac{d^m}{dz^m}P_\ell(z) \\ P_\ell^{-m}(z) = (-1)^m \frac{(\ell-m)!}{(\ell+m)!}P_\ell^m(z) \\ \ell, m \in \mathbb{N} \cup \{0\}, \quad m \le \ell, \quad z \in [-1,1] \end{gathered} \tag{3.5}$$

where P_ℓ - Legendre Polynomials.

Legendre Polynomials:

$$P_0(z) = 1, \quad P_\ell(z) = \frac{1}{2^\ell \ell!}\frac{d^\ell(z^2-1)^\ell}{dz^\ell}, \quad \ell \in \mathbb{N}, \quad z \in [-1,1] \tag{3.6}$$

3.1.2 Periodic Convolution: precursor

$$F^{(out)}_{a\ell_{in}\ell jm} = \sum_{b\in N(a)} F^{(in)}_{g(a,b)\ell_{in}j} K_{\ell m} \tag{3.7}$$

where $N(a)$ - neighborhood of a, $g(a,b) \to b^*$, F - features, K - filters.

After convolution number of indices increased. In order to mitigate it, equivariance preserving contraction should be done:

$$F^{(out)}_{a\ell_{out}i} = \sum_{b\in N(a)} \sum_{\ell_{in}} \sum_{j=-\ell_{in}}^{\ell_{in}} \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \sum_{m=-\ell}^{\ell} C_{\ell_{out}\ell_{in}\ell ijm} F^{(in)}_{g(a,b)\ell_{in}j} K_{\ell m} \tag{3.8}$$

where C - coupling coefficients (see Appendix A).

Note: rotational orders ℓ_{in}, ℓ_{out} are fixed by the shape of input and the required shape of output respectively, hence ℓ should be adjusted to satisfy the equation 3.8.

3.1.3 Periodic Convolution and Kernel

In the original implementation, Periodic Convolution is different from what given in equation 3.8. Coupling coefficients and sum over ℓ, m detach from Periodic Convolution and combine with filters to form another operation called Kernel. With normalization and multiplicities (multiple copies of the same rotation order) introduced, it results in the following.

Periodic Convolution:

$$F^{(n+1)}_{a(\ell_{out}ui)} = \frac{1}{\sqrt{|N(a)|}} \sum_{b\in N(a)} \sum_{(l_{in}vj)} F^{(n)}_{g(a,b)(\ell_{in}vj)} K_{ab\ell_{out}\ell_{in}uivj} \tag{3.9}$$

Kernel:

$$K_{ab\ell_{out}\ell_{in}uivj} = W_{\ell_{out}\ell_{in}ab} \sum_{(\ell m)} C_{\ell_{out}\ell_{in}\ell ijm} Y_{\ell mab} R_{ab\ell_{out}\ell_{in}uv\ell} \tag{3.10}$$

$F^{(n)}$ - features after layer n;
$F^{(0)}$ - input features;
C - coupling coefficients (see Appendix A);
W - normalization coefficients (see Equation B.36);
Y - real spherical harmonics;
R - Radial Basis Model output;
$|N(a)|$ - size of the set $N(a)$;
$N(a)$ - set of neighbors of atom a;
v - input multiplicity;
u - output multiplicity;
$j \in [0, 2\ell_{in}+1]$;
$i \in [0, 2\ell_{out}+1]$;
a - index of an atom;
b - index of a neighbor atom.

$$\sum_{(\ell_{in} v j)} \equiv \sum_{\ell_{in}} \sum_{v=0}^{k(\ell_{in})} \sum_{j=0}^{2\ell_{in}+1}$$

$$\sum_{(\ell m)} \equiv \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \sum_{m=0}^{2\ell+1}$$

Sets of ℓ_{out} and ℓ_{in} are defined per layer. ℓ_{out} for one layer is a ℓ_{in} for next. Each ℓ_{out}, ℓ_{in} is equipped with user defined output u and input v multiplicity respectively.

3.1.4 Gate

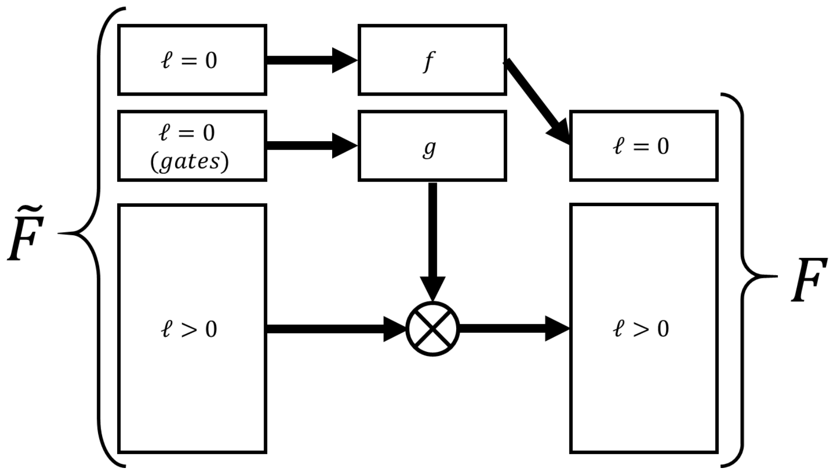

FIGURE 3.1: Gate

$$F_\ell = \begin{cases} f\left(\tilde{F}_\ell\right) & \text{if } \ell = 0 \\ g\left(\tilde{F}_{\ell=0}^{(gates)}\right)\tilde{F}_\ell & \text{if } \ell > 0 \end{cases} \tag{3.11}$$

f, g are some non-linearities. For scalars ($\ell = 0$) non-linearity can be applied directly without breaking equivariance. For higher order components ($\ell > 0$), a non-linearity can be applied by operating on the vector norm, which is invariant to rotation, or "gating" the higher order components with a separate scalar feature. Later is used. Each rotational order ℓ among $\ell > 0$ for all components m, gets **one** corresponding scalar feature ($\ell = 0(gates)$) as a multiplicative coefficient.

3.2 Issues of the original implementation

3.2.1 Real spherical harmonics: serialization

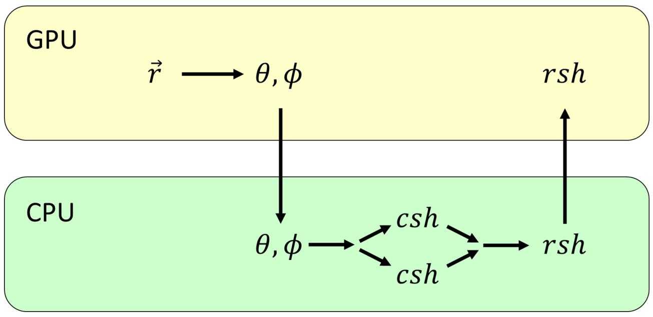

FIGURE 3.2: Original routine for real spherical harmonics.
`csh` - complex spherical harmonic, `rsh` - real spherical harmonic

Compute times (per call) are provided for the relative comparison:

- Angles conversion, GPU-CPU and CPU-GPU transfers, post-processing: 0.5 ms
- Computing `csh`: $2\times$ 3.55 ms $=$ 7.1 ms (recursive sequential implementation)
- Combining csh into rsh: 1.0 ms

Total: 8.7 ms (per call).

The main issue is underused parallelism. The real spherical harmonics routine is usually requested for pack of thousands of entries - the backbone `csh` routine applies essentially the same calculation process to all entries one-by-one.

3.2.2 Kernel to Periodic Convolution: memory bottleneck

The kernel output $K_{ab\ell_{out}\ell_{in}uivj}$ dominates memory usage.

Consider:

- $|a| = 100$ atoms in a structure
- $|b| = 12$ neighbor atoms for each atom a
- $\ell_{out} = \ell_{in} = [0, 1, 2, 3, 4]$ - rotational orders of output/input representations
- $|u| = |v| = 16$ - multiplicities for each rotational order
- $|i| = 2\ell_{out} + 1$ and $|j| = 2\ell_{in} + 1$ - number of components per rotational order
- 64-bit entries - rounding error at 32-bit breaks equivariance after a couple of layers

When all numbers are plugged in, the size of K appears to be **1.2 GB**. Keep in mind that in train mode PyTorch saves it to the buffer for backward pass on each layer!

In comparison, the next largest (and unavoidable) object is Radial Basis Model output $R_{ab\ell_{out}\ell_{in}uv\ell}$ under the same considerations requires just **152 MB**.

3.2.3 Kernel to Periodic Convolution: serialized operations

All input tensors in Equations 3.9 and 3.10, aside for W, have varying length for one or more logical indices. Padding leads to the quite dramatic memory overhead. At the same time, PyTorch does not support batching operations with varying rules within a batch. Therefore, in original implementation contingent blocks (for which operations can actually be vectorized) get picked one by one in a loop (serialization), which results in low GPU utilization.

3.2.4 Layer layout: delayed garbage collection

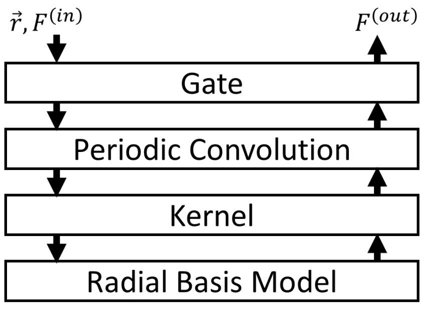

FIGURE 3.3: Original (nested) layer

In Python, garbage collection depends on link counting. When number of links to a variable reaches zero, underlying memory gets released. Intermediate variables created within a function have zero links once function ends, unless explicitly deleted. Nested layout unnecessary extends the lifetime of those no longer used variables, which results in higher peak memory usage.

3.3 Solutions

TABLE 3.1: Effect of the implemented solution on the execution time

Time is given for one training epoch on full MP database.
Specifications for machine can be found in supplementary materials.

Implemented solution (cumulative)	**Execution time,** hours per epoch
Original implementation	12 (estimate)
Merged Periodic Convolution and Kernel (Python)	5
Custom CUDA implementation for real spherical harmonics	4
Merged Periodic Convolution and Kernel (CUDA) + sequential layer layout	1.5

3.3.1 Real spherical harmonics: custom CUDA implementation

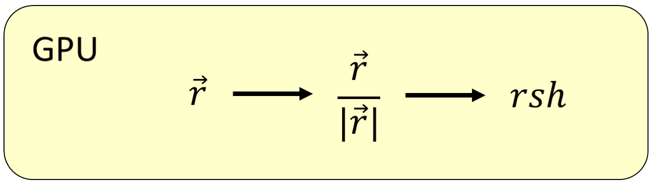

FIGURE 3.4: New routine for real spherical harmonics

We do not need spherical harmonics for large rotation orders ℓ. That allows to by-pass recursive formula. I used online tables with explicit expressions (no recursion) [39]: for real spherical harmonics in Cartesian basis ($\ell \leq 4$), and for complex spherical harmonics in Spherical basis ($5 \leq \ell \leq 10$) to construct list of explicit expressions in Cartesian basis ($\ell \leq 10$). Those I coded in the CUDA extension that fully utilizes paralleling over the inputs.

Compute times (per call):

- Input normalization, post-processing: 0.18 ms
- Computing rsh: 0.0076 ms - more than 1000x faster than prior backbone routine

Total: 0.19 ms (per call) - overall speedup is 46.8x.

Tricks of the trade. The (fixed) coefficients evaluation requires square roots and divisions. 'constexpr' forced it from the run time (recalculate on each execution) to the compile time (calculate once and embed the result as a part of the code), so that the only arithmetic operations in the remaining routine comprises are multiplication and addition - a major boost to the performance.

3.3.2 Merge Kernel and Periodic Convolution

To avoid the issue of stashing Kernel outputs K (described in 3.2.2) we combine Kernel and Periodic Convolution into the single operation. The single operation from the viewpoint of PyTorch. This requires the `backward()` operation for calculation of gradients with a custom custom operation (see C.2, C.3 for derivation). Otherwise, the auto-generated computation graph in PyTorch will contain many nodes corresponding to small operations, each equipped with its own buffer, cumulatively leading to the very same memory issue.

In Ref. [18], Ben Miller addresses the problem in a similar way for the full Convolution (all-to-all interactions between atoms in a structure). Although, keep in mind that it is inapplicable to Materials Project Database as structures in it are relatively large - even optimized full Convolution will result in out-of-memory exception.

3.3.3 Kernel and Periodic Convolution: CUDA implementation

See Appendix C for the details of implementation.

3.3.4 Layer layout: nested to sequential

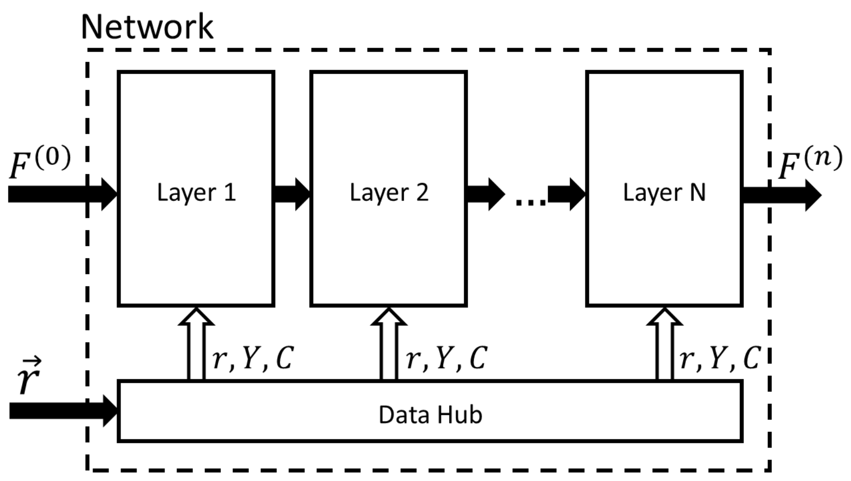

FIGURE 3.5: New network layout

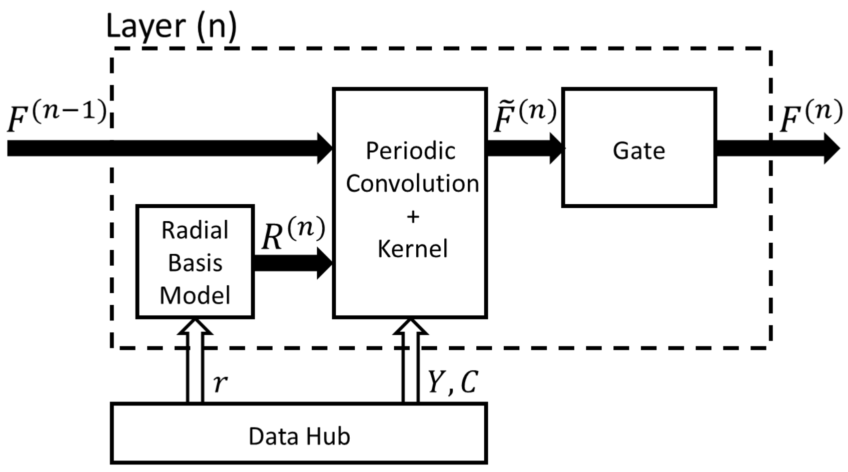

FIGURE 3.6: New (sequential) layer

The purpose of the introduced Data Hub block is to preprocess and store variables that are going to be reused in the network. Those variables can be attributed either to the network or to the input. Network-lifetime variables: coupling coefficients C and memory offsets (see Appendix C) are calculated once when network is instantiated and never change. Input-lifetime variables: absolute distances r and real spherical harmonics Y are replaced once new input is supplied, but reused within a network across the layers (for both forward and backward passes).

Chapter 4

Elastic Tensor

Euclidean Neural Networks support training on tensors of an arbitrarily high rank while preserving rotation equivariance. It is most common to express geometric tensors of physical system as Cartesian tensors. However, E(3)NNs operate on geometric tensors in the irreducible basis. Therefore, it is necessary to express tensor in irrep components that in the case of the elastic tensor can be associated with specific real spherical harmonics.

In this chapter, I derive forward and inverse transformations between Cartesian coordinates and components of real spherical harmonics (which transform as the irreps of $SO(3)$) for the elastic tensor.

4.1 Definition

The elastic tensor (also called elasticity tensor) is a rank-4 tensor that quantifies the response of a body to deformations. In particular, it appears in Hooke's Law for continuous media [8]:

$$\sigma_{ij} = C_{ijkn}\varepsilon_{kn} \qquad i,j,k,n \in \{1,2,3\} \tag{4.1}$$

where σ - stress tensor; ε - strain tensor; C - elastic tensor, $\{1,2,3\} \equiv \{x,y,z\}$.

For comparison, Hooke's Law for linear spring [40]:

$$F = k\Delta x \tag{4.2}$$

where F - counteracting force; k - elasticity coefficient; Δx - displacement from equilibrium position.

The elastic tensor C has the following symmetries [8]:

$$\sigma_{ij} = \sigma_{ji} \qquad \varepsilon_{kn} = \varepsilon_{nk} \qquad C_{ijkn} = C_{knij} = C_{jikn} = C_{ijnk} \tag{4.3}$$

Consequently, elastic tensor has at most 21 independent elements out of 81 that an arbitrary rank-4 tensor can posses. This implies that equation 4.1 is redundant. One of the approaches to partially reduce redundancy is Voigt notation, hereby is the variation used in MP [41, 42].

In the compact form:

$$\tilde{\sigma}_i = \tilde{C}_{ij}\tilde{\varepsilon}_j \qquad i,j \in \{1,\ldots 6\} \tag{4.4}$$

where mapping for indices: $11 \to 1, 22 \to 2, 33 \to 3, 23 \to 4, 13 \to 5, 12 \to 6$.

In the full form:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16}\\ C_{12} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26}\\ C_{13} & C_{23} & C_{33} & C_{34} & C_{35} & C_{36}\\ C_{14} & C_{24} & C_{34} & C_{44} & C_{45} & C_{46}\\ C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & C_{56}\\ C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66}\end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ 2\varepsilon_{23}\\ 2\varepsilon_{13}\\ 2\varepsilon_{12}\end{bmatrix} \tag{4.5}$$

4.2 Decompositions (literature)

Depending on the type of lattice system crystal can have additional point symmetries and consequently less than 21 independent components [8]. I focus only on the most general case - triclinic lattice system.

There are multiple ways to decompose the elastic tensor. In particular, Ref. [43] shows decomposition to and reconstruction from harmonic tensors. The exact correspondence between harmonic tensors and spherical harmonic tensors is not apparent (they are not the same thing!). From Ref. [44], we know that there should be two copies of $\ell = 0$, two copies of $\ell = 2$ and one of $\ell = 4$ irreps.

Ref. [45] presents the decomposition of the elastic tensor in **complex** spherical harmonic components, as well as the reconstruction, appears to be the closest match. Whats left is adapting the approach to **real** spherical harmonics.

4.3 Real spherical harmonic decomposition

Following Ref. [45], conversion from Cartesian coordinates (i, j, k, n) to covariant components $(\alpha, \beta, \gamma, \delta)$ for elastic tensor C:

$$C_{\alpha\beta\gamma\delta} = C_{ijkn} K_{\alpha i} K_{\beta j} K_{\gamma k} K_{\delta n} \tag{4.6}$$

and inverse:

$$C_{ijkn} = C_{\alpha\beta\gamma\delta} J_{i\alpha} J_{j\beta} J_{k\gamma} J_{n\delta} \tag{4.7}$$

where matrices K, J are normalized permutation matrices (see Appendix E.1):

$$K = -\sqrt{\frac{3}{4\pi}} \begin{bmatrix} 0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix} \qquad J = -\sqrt{\frac{4\pi}{3}} \begin{bmatrix} 0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} = \frac{4\pi}{3} K^T \tag{4.8}$$

Reconstruction equation for elastic tensor in covariant components, indices $(\alpha\beta\gamma\delta)$ are implied for C, A, B, D, E, H:

$$C = A q_{00} + \sum_{m=-2}^{2} B_m q_{2m} + D s_{00} + \sum_{m=-2}^{2} E_m s_{2m} + \sum_{m=-4}^{4} H_m s_{4m} \tag{4.9}$$

where $q_{00}, q_{2m}, s_{00}, s_{2m}, s_{4m}$ are real spherical harmonic components.

A, B_m, D, E_m, H_m are fixed coefficients calculated as follows (see Appendix E.2 for derivation):

$$A^{(\alpha\beta\gamma\delta)} = G^{0,1,1}_{0,\alpha,\beta} G^{0,1,1}_{0,\gamma,\delta} G^{0,0,0}_{0,0,0} \tag{4.10}$$

$$B_m^{(\alpha\beta\gamma\delta)} = \left(G^{0,1,1}_{0,\alpha,\beta} G^{2,1,1}_{m,\gamma,\delta} + G^{0,1,1}_{0,\gamma,\delta} G^{2,1,1}_{m,\alpha,\beta} \right) G^{0,2,2}_{0,m,m} \tag{4.11}$$

$$D^{(\alpha\beta\gamma\delta)} = \sum_{m'=-2}^{2} G^{2,1,1}_{m',\alpha,\beta} G^{2,1,1}_{m',\gamma,\delta} G^{0,2,2}_{0,m',m'} \tag{4.12}$$

$$E_m^{(\alpha\beta\gamma\delta)} = \sum_{m_1=-2}^{2} \sum_{m_2=-2}^{2} G^{2,1,1}_{m_1,\alpha,\beta} G^{2,1,1}_{m_2,\gamma,\delta} G^{2,2,2}_{m,m_1,m_2} \tag{4.13}$$

$$H_m^{(\alpha\beta\gamma\delta)} = \sum_{m_1=-2}^{2} \sum_{m_2=-2}^{2} G^{2,1,1}_{m_1,\alpha,\beta} G^{2,1,1}_{m_2,\gamma,\delta} G^{4,2,2}_{m,m_1,m_2} \tag{4.14}$$

where G are the real Gaunt (R-Gaunt) coefficients [46, 47] (see Appendix D).

The decomposition equation is an inverse of the Equation 4.9, the easiest way to obtain it is to employ matrix multiplication. Let C, A, B, D, E, H be flatten into column-vectors over indices $(\alpha\beta\gamma\delta)$, then:

$$C = [A, B_m, D, E_m, H_m] \begin{bmatrix} q_{00} \\ q_{2m} \\ s_{00} \\ s_{2m} \\ s_{4m} \end{bmatrix} = TS \tag{4.15}$$

C has shape $[81 \times 1]$, T $[81 \times 21]$, S $[21 \times 1]$.

Due to its index permutation symmetries C contains identical entries. Corresponding rows of T are identical as well. It is possible to select a subset of linearly independent components (see Appendix E.3) such that:

$$\hat{C} = \hat{T} S \tag{4.16}$$

where hat denotes subset, $\hat{C}$ has shape $[21 \times 1]$, $\hat{T}$ $[21 \times 21]$, S $[21 \times 1]$.

$\hat{T}$ is a full rank matrix, meaning that decomposition equation is simply:

$$S = \hat{T}^{-1} \hat{C} \tag{4.17}$$

Exact values of matrices $\left(8\pi^{\frac{3}{2}}\right) \hat{T}$ and $\left(\frac{1}{8\pi^{\frac{3}{2}}}\right) \hat{T}^{-1}$ are provided in supplementary materials.

Chapter 5

Conclusion

5.1 Contribution

In this thesis, I presented the following contributions: First, I performed an analysis over Materials Project Database, which is the first analysis on this database as such (that I am aware of). It revealed that choice of radial cutoff and number of layers in the network do not have to be arbitrary - data gives strong prior. For Materials Project Database it is 5 Å and 4 layers respectively. Second, I contributed the optimized CUDA implementation for `Periodic Convolution` in `e3nn` that enables scaling to real-world scenarios. And finally, I derived decomposition and reconstruction transformations for elastic tensor, so that it can be used as a prediction target in `e3nn`.

5.2 Future work

There are two main vectors of the further development: engineering and theory.

On the side of engineering. Euclidean Neural Networks fit nicely within a theoretical framework of message passing neural networks (MPNN) [9] and can benefit from results of ongoing research in the field of graph neural networks. However, as it currently is, my optimized CUDA extensions are not compatible with a graph neural network software frameworks that implement MPNN, such as pytorch-geometic [48] and DGL [49], hence it is a priority to make respective adjustments.

As for the theory. Procedure described in Chapter 4 and Ref. [45] for elastic tensor is not general. In a sense that direct transfer of it is not valid just for any tensor (e.g. it is not valid for piezoelectric tensor $d_{i(jk)}$). It would be desirable for future work to extend these procedures to arbitrary rank tensors with various additional index permutation symmetries.

Appendix A

E(3)NN: Coupling coefficients

(Credit for the derivation presented in this appendix belongs to Mario Geiger)

In order for the feature tensor to not increase number of indices on each layer of a network (see Equation 3.7), an equivariant contraction operation is required. Purpose of this appendix is to show how to obtain coefficients C for a contraction operation of a given form such that operation is indeed equivariant. Throughout this appendix, Einstein summation notation is used (repeated indices are summed over).

$$Z_{\ell_{out} i} = C_{\ell_{out} \ell_{in} \ell i j m} F_{\ell_{in} j} K_{\ell m} \tag{A.1}$$

where F, K, Z - irrep tensors (e.g. input, kernels, output); C - coupling coefficients.

Irrep tensors act under proper rotations in the same way as real spherical harmonics. Under some rotation of space, Equation A.1:

$$D^{\ell_{out}}_{ii'} Z_{\ell_{out} i'} = C_{\ell_{out} \ell_{in} \ell i j m} D^{\ell_{in}}_{jj'} F_{\ell_{in} j'} D^{\ell}_{mm'} K_{\ell m'} \tag{A.2}$$

where $D \equiv D(\Re(\delta\Omega))$ - rotation matrix over real spherical harmonics [37]. Note that D is a square matrix, and upper script (it is not an index per se) defines the size.

Substitute A.1 into A.2 and rearrange:

$$D^{\ell_{out}}_{ii'} C_{\ell_{out} \ell_{in} \ell i' k n} F_{\ell_{in} k} K_{\ell n} = D^{\ell_{in}}_{jj'} D^{\ell}_{mm'} C_{\ell_{out} \ell_{in} \ell i j m} F_{\ell_{in} j'} K_{\ell m'} \tag{A.3}$$

From construction, if coupling coefficients C abide Equation A.3, then equivariance is retained.

From orthonormality of D follows:

$$D^{\ell_{out}}_{ii^*} D^{\ell_{out}}_{ii'} = \delta_{i^* i'} \tag{A.4}$$

where δ - Kronecker's delta.

Contraction of both sides of A.3 with A.4 gives:

$$C_{\ell_{out} \ell_{in} \ell i^* k n} F_{\ell_{in} k} K_{\ell n} = D^{\ell_{out}}_{ii^*} D^{\ell_{in}}_{jj'} D^{\ell}_{mm'} C_{\ell_{out} \ell_{in} \ell i j m} F_{\ell_{in} j'} K_{\ell m'} \tag{A.5}$$

In A.5, relabel $k \to j'$, $n \to m'$, $i^* \to i'$ and match sides, then:

$$C_{\ell_{out} \ell_{in} \ell i' j' m'} = D^{\ell_{out}}_{ii'} D^{\ell_{in}}_{jj'} D^{\ell}_{mm'} C_{\ell_{out} \ell_{in} \ell i j m} \tag{A.6}$$

Hence, valid values of $C_{\ell_{out} \ell_{in} \ell i j m}$ belong to null-space of $\left(I - D^{\ell_{out}}_{ii'} D^{\ell_{in}}_{jj'} D^{\ell}_{mm'} \right)$, where I - identity matrix, and $D^{\ell_{out}}_{ii'} D^{\ell_{in}}_{jj'} D^{\ell}_{mm'}$ gets reshaped into square matrix $(ijm) \times (i'j'm')$.

Appendix B

E(3)NN: Normalization

In this appendix, I derive expression for normalization coefficients W such that every component of output features has unit variance. In Ref. [18], Ben Miller with help of Mario Geiger derives W based on integration over initial distribution of network weights. I am going to use integration over the space instead, with operations defined as follows:

Mean:

$$\langle X \rangle \equiv \mu \equiv \langle X \rangle_{\theta,\phi,r} \equiv \frac{1}{4\pi}\frac{3}{r_{cut}^3}\int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi}\int_{r=0}^{r_{cut}} Xr^2 \sin\theta \, d\theta \, d\phi \, dr \tag{B.1}$$

Variance:

$$Var(X) \equiv \sigma^2 \equiv \langle X^2 \rangle - \langle X \rangle^2 \tag{B.2}$$

Covariance:

$$Cov(X,Z) \equiv \langle XZ \rangle - \langle X \rangle \langle Z \rangle \tag{B.3}$$

Additionally, I am going to review special cases stemming from periodicity of crystals. For those purposes, it is convenient to use division into Periodic Convolution and Kernel, as given by equations 3.9, 3.10.

$$F^{(n+1)}_{a(\ell_{out}ui)} = \frac{1}{\sqrt{|N(a)|}} \sum_{b\in N(a)} \sum_{(l_{in}vj)} F^{(n)}_{g(a,b)(\ell_{in}vj)} K_{ab\ell_{out}\ell_{in}uivj} \tag{B.4}$$

$$K_{ab\ell_{out}\ell_{in}uivj} = W_{\ell_{out}\ell_{in}ab} \sum_{(\ell m)} C_{\ell_{out}\ell_{in}\ell ijm} Y_{\ell mab} R_{ab\ell_{out}\ell_{in}uv\ell} \tag{B.5}$$

I drop non-essential indices to avoid visual clutter.
$F^{(n+1)} \to F'$, $F^{(n)} \to F$, $(\ell_{in}vj) \to j'$

$$F'_a = \frac{1}{\sqrt{|N(a)|}} \sum_{b\in N(a)} \sum_{j'} F_{g(a,b)j'} K_{abj'} \tag{B.6}$$

$$K = W \sum_{(\ell m)} C_{\ell m} Y_{\ell m} R_{\ell} \tag{B.7}$$

B.1 Kernel

B.1.1 Mean

$$\langle K \rangle = \left\langle W \sum_{(\ell m)} C_{\ell m} Y_{\ell m} R_\ell \right\rangle = W \sum_{(\ell m)} C_{\ell m} \langle Y_{\ell m} R_\ell \rangle \tag{B.8}$$

$$\langle Y_{\ell m} R_\ell \rangle \equiv \langle Y_{\ell m}(\theta,\phi) R_\ell(r) \rangle_{\theta,\phi,r} = \langle Y_{\ell m}(\theta,\phi) \rangle_{\theta,\phi} \langle R_\ell(r) \rangle_r \tag{B.9}$$

$$\langle Y_{\ell m}(\theta,\phi) \rangle_{\theta,\phi} \equiv \frac{1}{4\pi} \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} Y_{\ell m}(\theta,\phi) \sin\theta d\theta d\phi = \begin{cases} \frac{1}{2\sqrt{\pi}}, & \text{if } \ell = 0, m = 0 \\ 0, & \text{otherwise} \end{cases} \tag{B.10}$$

$$\langle R_\ell(r) \rangle_r = \frac{3}{r_{cut}^3} \int_{r=0}^{r_{cut}} R_\ell(r) r^2 dr \equiv \mu_{R_\ell} \tag{B.11}$$

Substitute B.11 and B.10 into B.9:

$$\langle Y_{\ell m} R_\ell \rangle = \frac{\mu_{R_0}}{2\sqrt{\pi}} [\ell = 0][m = 0] \tag{B.12}$$

$[\ell = 0]$ - Iverson notation: if condition in the bracket True, then 1, otherwise 0.

Substitute B.12 into B.8:

$$\mu_{K_\ell} = \langle K_\ell \rangle = W \frac{C_{00} \mu_{R_0}}{2\sqrt{\pi}} [\ell = 0] \tag{B.13}$$

Assertion 1. $\mu_{R_0} = 0$. We can force all μ_{R_ℓ} and $\sigma^2_{R_\ell}$. As for now, only μ_{R_0} being zero is important, otherwise further expressions for variances jump in complexity.

$$\langle Y_{\ell m} R_\ell \rangle = 0 \tag{B.14}$$

$$\mu_K = 0 \tag{B.15}$$

B.1.2 Variance

$$\begin{aligned} Var(K) &= Var\left(W \sum_{(\ell m)} C_{\ell m} Y_{\ell m} R_\ell \right) = \\ &= W^2 Var\left(\sum_{(\ell m)} C_{\ell m} Y_{\ell m} R_\ell \right) = \\ &= W^2 \sum_{(\ell m)} Var\left(C_{\ell m} Y_{\ell m} R_\ell \right) + 2W^2 \sum_{1 \le i < j \le n} Cov\left(C_i Y_i R_{f(i)}, C_j Y_j R_{f(j)} \right) = \\ &= W^2 \sum_{(\ell m)} C_{\ell m}^2 Var\left(Y_{\ell m} R_\ell \right) + 2W^2 \sum_{1 \le i < j \le n} C_i C_j Cov\left(Y_i R_{f(i)}, Y_j R_{f(j)} \right) \end{aligned} \tag{B.16}$$

All transformations rely on general variance and covariance properties without any further assumptions. Indices i, j go through the pairs (ℓm), $f(...)$ accounts for the fact that R does not have index m.

Using B.14:

$$Var\left(Y_{\ell m}R_\ell\right) = \langle (Y_{\ell m}R_\ell)^2\rangle - \langle Y_{\ell m}R_\ell\rangle^2 = \langle Y_{\ell m}^2\rangle_{\theta,\phi}\langle R_\ell^2\rangle_r \tag{B.17}$$

From orthonormality of real spherical harmonics follows:

$$\left\langle Y_{\ell m}^2(\theta,\phi)\right\rangle_{\theta,\phi} \equiv \frac{1}{4\pi}\int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi} Y_{\ell m}^2(\theta,\phi)\sin\theta d\theta d\phi = \frac{1}{4\pi} \tag{B.18}$$

Caveat: equality B.18 is only valid if $r \neq 0$. If $r = 0$, then angles θ, ϕ are not well defined and value should set to be zero for all ℓ, m except for $\ell = m = 0$. This is going to be crucial for self-interactions.

$$\langle R_\ell^2\rangle_r = \langle R_\ell^2\rangle_r - \langle R_\ell\rangle_r^2 + \langle R_\ell\rangle_r^2 \equiv \sigma_{R_\ell}^2 + \mu_{R_\ell}^2 \tag{B.19}$$

Substitute B.18 and B.19 into B.17:

$$Var\left(Y_{\ell m}R_\ell\right) = \frac{1}{4\pi}\left(\sigma_{R_\ell}^2 + \mu_{R_\ell}^2\right) \tag{B.20}$$

$$\begin{aligned} Cov\left(Y_iR_{f(i)}, Y_jR_{f(j)}\right) &= \langle Y_iY_jR_{f(i)}R_{f(j)}\rangle - \langle Y_iR_{f(i)}\rangle\langle Y_jR_{f(j)}\rangle = \\ &= \langle Y_iY_j\rangle_{\theta,\phi}\langle R_{f(i)}R_{f(j)}\rangle_r - \langle Y_i\rangle_{\theta,\phi}\langle Y_j\rangle_{\theta,\phi}\langle R_{f(i)}\rangle_r\langle R_{f(j)}\rangle_r \end{aligned} \tag{B.21}$$

Recall that $i \neq j$, therefore from orthogonality $\langle Y_iY_j\rangle_{\theta,\phi} = 0$, and $\langle Y_i\rangle_{\theta,\phi}\langle Y_j\rangle_{\theta,\phi} = 0$, because at most one of i, j can correspond to $\ell = 0, m = 0$, which is the only non-zero option for $\langle Y_{\ell m}\rangle$, therefore:

$$Cov\left(Y_iR_{f(i)}, Y_jR_{f(j)}\right) \equiv 0 \tag{B.22}$$

Substitute B.22 and B.20 into B.16:

$$Var(K) \equiv \sigma_K^2 = \frac{W^2}{4\pi}\sum_{(\ell m)} C_{\ell m}^2\left(\sigma_{R_\ell}^2 + \mu_{R_\ell}^2\right) \tag{B.23}$$

B.2 Periodic Convolution

Features are attached to the space. They implicitly depend on coordinates through index $g(a,b)$.

Assertion 2. Input features have zero mean and unit variance. We can always force it on the first layer.

$$\mu_F = 0, \qquad \sigma_F^2 = 1 \tag{B.24}$$

B.2.1 Mean

From linearity:

$$\begin{aligned} \langle F'_a \rangle &= \left\langle \frac{1}{\sqrt{|N(a)|}} \sum_{b \in N(a)} \sum_{j'} F_{g(a,b)j'} K_{abj'} \right\rangle = \\ &= \frac{1}{\sqrt{|N(a)|}} \sum_{b \in N(a)} \sum_{j'} \left\langle F_{g(a,b)j'} K_{abj'} \right\rangle \end{aligned} \tag{B.25}$$

Assertion 3. Features and Kernel outputs are linearly independent.

Using B.15 or B.24:

$$\langle F'_a \rangle = \frac{1}{\sqrt{|N(a)|}} \sum_{b \in N(a)} \sum_{j'} \left\langle F_{g(a,b)j'} \right\rangle \left\langle K_{abj'} \right\rangle = 0 \tag{B.26}$$

$\mu_F = 0$ pertains throughout the layers.

B.2.2 Variance

Assertion 4. Pair of products of feature and Kernel output are linearly independent across index $j' \equiv (\ell_{in} v j)$, and across index b unless $g(a,b_1) = g(a,b_2)$.

$$\begin{aligned} Var(F'_a) &= Var\left(\frac{1}{\sqrt{|N(a)|}} \sum_{b \in N(a)} \sum_{j'} F_{g(a,b)j'} K_{abj'} \right) = \\ &= \frac{1}{|N(a)|} Var\left(\sum_{b \in N(a)} \sum_{j'} F_{g(a,b)j'} K_{abj'} \right) = \\ &= \frac{1}{|N(a)|} Var\left(\sum_{j'} \sum_{b \in N(a)} F_{g(a,b)j'} K_{abj'} \right) = \\ &= \frac{1}{|N(a)|} \sum_{j'} Var\left(\sum_{b \in N(a)} F_{g(a,b)j'} K_{abj'} \right) \end{aligned} \tag{B.27}$$

I consider two extreme cases:

1. $g(a,b)$ are different for all $b \in N(a)$ for a fixed a
2. $g(a,b)$ are the same for all $b \in N(a)$ for a fixed a

Case 1: all neighbor atoms are different

$g(a,b)$ are different for all $b \in N(a)$ for a fixed a - this always holds for molecules. As for the crystals, it is sufficient, but not generally necessary to have radial cutoff smaller than the radius of the largest sphere that can fit within the unit cell.

Using assertions #4, #3, #2 and B.15, B.23:

$$\begin{aligned} Var\left(\sum_{b\in N(a)} F_{g(a,b)j'} K_{abj'}\right) &= \sum_{b\in N(a)} Var\left(F_{g(a,b)j'} K_{abj'}\right) = \\ &= \sum_{b\in N(a)} \left(\mu_F^2\sigma_K^2 + \mu_K^2\sigma_F^2 + \sigma_F^2\sigma_K^2\right) = \\ &= \sum_{b\in N(a)} \left(\mu_K^2 + \sigma_K^2\right) = \\ &= |N(a)| \left(\mu_K^2 + \sigma_K^2\right) = \\ &= |N(a)| \frac{W^2}{4\pi} \sum_{(\ell m)} C_{\ell m}^2 \left(\sigma_{R_\ell}^2 + \mu_{R_\ell}^2\right) \end{aligned} \tag{B.28}$$

Substitute B.28 into B.27:

$$Var(F_a') = \frac{W^2}{4\pi} \sum_{j'} \sum_{(\ell m)} C_{\ell m}^2 \left(\sigma_{R_\ell}^2 + \mu_{R_\ell}^2\right) \tag{B.29}$$

Assertion 5. for all ℓ, j': $\sigma_{R_\ell}^2 \equiv \sigma_R^2$, $\mu_{R_\ell} \equiv \mu_R$. This can be forces.

$$Var(F_a') = \frac{W^2}{4\pi} \left(\sigma_R^2 + \mu_R^2\right) \sum_{j'} \sum_{(\ell m)} C_{\ell m}^2 \tag{B.30}$$

Coupling coefficients C appear to abide the same summation identity as Wigner-3j symbol. For the lack of theoretical proof, I numerically verified following conjecture for all valid combinations of $\ell_{out} \in \{0, \dots 10\}$, $\ell_{in} \in \{0, \dots 10\}$, $\ell \in \{0, \dots 20\}$.

$$\begin{aligned} \sum_{j'} \sum_{(\ell m)} C_{\ell m}^2 \equiv \sum_{(\ell_{in} v j)} \sum_{(\ell m)} C_{\ell_{out}\ell_{in}\ell ijm}^2 &= \\ = \sum_{\ell_{in}} \sum_{v=0}^{k(\ell_{in})} \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \sum_{j=-\ell_{in}}^{\ell_{in}} \sum_{m=-\ell}^{\ell} \left(C_{\ell_{out}\ell_{in}\ell ijm}\right)^2 &= \\ = \sum_{\ell_{in}} \sum_{v=0}^{k(\ell_{in})} \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \frac{1}{2\ell_{out}+1} &= \\ = \sum_{\ell_{in}} \sum_{v=0}^{k(\ell_{in})} \frac{1+2min(\ell_{out}, \ell_{in})}{2\ell_{out}+1} &= \\ = \frac{1}{2\ell_{out}+1} \sum_{\ell_{in}} \sum_{v=0}^{k(\ell_{in})} \left(1+2min(\ell_{out}, \ell_{in})\right) & \end{aligned} \tag{B.31}$$

$$Var(F'_a) = \frac{W^2}{4\pi}\left(\sigma_R^2 + \mu_R^2\right)\frac{1}{2\ell_{out}+1}\sum_{\ell_{in}}\sum_{v=0}^{k(\ell_{in})}\left(1 + 2min(\ell_{out}, \ell_{in})\right) \tag{B.32}$$

Setting $Var(F'_a) = 1$ and solving for W gives:

$$W = \frac{\sqrt{4\pi}\sqrt{2\ell_{out}+1}}{\sqrt{\sigma_R^2+\mu_R^2}\sqrt{\sum_{\ell_{in}}\sum_{v=0}^{k(\ell_{in})}\left(1+2min(\ell_{out},\ell_{in})\right)}} \tag{B.33}$$

Notice that for $\mu_R = 0, \sigma_R^2 = 1$, expression turns out to be the same as in the Ref. [18]:

$$\boxed{W_{r\neq 0} = \frac{\sqrt{4\pi}\sqrt{2\ell_{out}+1}}{\sqrt{\sum_{\ell_{in}}\sum_{v=0}^{k(\ell_{in})}\left(1+2min(\ell_{out},\ell_{in})\right)}}} \tag{B.34}$$

Returning to the self-interactions ($r = 0$), mentioned caveat results in $C_{\ell m} \to C_{00}$, changing the sum:

$$\boxed{W_{r=0} = \frac{\sqrt{4\pi}\sqrt{2\ell_{out}+1}}{\sqrt{\sum_{\ell_{in}}\sum_{v=0}^{k(\ell_{in})}(1)}}} \tag{B.35}$$

Combined:

$$\boxed{W = W_{r\neq 0}[r \neq 0] + W_{r=0}[r = 0]} \tag{B.36}$$

Case 2: all neighbor atoms are the same (copies)

$g(a, b)$ are the same for all $b \in N(a)$ for a fixed a - this happens for example when unit cell contains atoms of the only one atom type.

Using assertion #2 and B.15:

$$\begin{aligned} Var\left(\sum_{b\in N(a)} F_{g(a,b)j'}K_{abj'}\right) &= Var\left(F_{b^*j'}\sum_{b\in N(a)} K_{abj'}\right) = \\ = \mu_F^2 Var\left(\sum_{b\in N(a)} K_{abj'}\right) &+ \sigma_F^2\left\langle\sum_{b\in N(a)} K_{abj'}\right\rangle^2 + \sigma_F^2 Var\left(\sum_{b\in N(a)} K_{abj'}\right) = \\ = Var\left(\sum_{b\in N(a)} K_{abj'}\right) \end{aligned} \tag{B.37}$$

I consider 3 subcases:

1. Independence
2. Destructive interference
3. Constructive interference

1. Kernel outputs K are independent over index b.

$$Var\left(\sum_{b\in|N(a)|} K_{abj'}\right) = \sum_{b\in|N(a)|} Var\left(K_{abj'}\right) = |N(a)|\sigma_K^2 \tag{B.38}$$

   Identical to the case 1.

2. Kernels perfectly counteract each other.

$$Var\left(\sum_{b\in|N(a)|} K_{abj'}\right) = 0 \tag{B.39}$$

   Consequently,

$$Var\left(F'_{\ell_{out}}\right) = 0 \tag{B.40}$$

   Combined with Eq. B.26 this means that signal completely vanishes.

3. Kernels perfectly align with each other.

$$Var\left(\sum_{b\in N(a)} K_{abj'}\right) = |N(a)|^2\sigma_K^2 \tag{B.41}$$

   Assuming normalization coefficient W as given by Eq. B.36, it yields:

$$Var\left(F'_{\ell_{out}}\right) = |N(a)| \geq 1 \tag{B.42}$$

   If this subcase is ever to occur practically, then it almost certainly is going to result in signal / gradients explosion - breaking the training.

Appendix C

Periodic Convolution with Kernel

In order to avoid huge memory peaks induced by the Kernel output (as outlined in Sections 3.2.2, 3.3.2), Periodic Convolution (Eq. 3.9) and Kernel (Eq. 3.10) operations should be merged into the following:

$$F^{(n+1)}_{a(\ell_{out}ui)} = \frac{1}{\sqrt{|N(a)|}} \sum_{b\in N(a)} \sum_{\ell_{in}} W_{\ell_{out}\ell_{in}ab} \sum_{(vj)} \sum_{(\ell m)} C_{\ell_{out}\ell_{in}\ell ijm} F^{(n)}_{g(a,b)(\ell_{in}vj)} \times \\ \times Y_{\ell mab} R_{ab\ell_{out}\ell_{in}\ell uv} \tag{C.1}$$

$F^{(n)}$ - features after layer n;
$F^{(0)}$ - input features;
C - coupling coefficients (see Appendix A);
W - normalization coefficients (see Appendix B);
$Y \equiv Y(\vec{r}/|\vec{r}|)$ - real spherical harmonic;
$R \equiv R(|\vec{r}|)$ - Radial Basis Model outputs;
$|N(a)|$ - size of the set $N(a)$;
$N(a)$ - set of neighbours of the atom a;
$v \equiv v(\ell_{in})$ - input multiplicity;
$u \equiv u(\ell_{out})$ - output multiplicity;
$j \in [0, 2\ell_{in} + 1]$;
$i \in [0, 2\ell_{out} + 1]$;
a - index of the origin atom;
b - index of the neighbor atom;
$g(a, b) \to b^*$ - map neighbor atom index to the index in the original unit cell;
$b^* \in [0, |a| - 1]$.

$$\sum_{(vj)} \equiv \sum_{v=0}^{k(\ell_{in})} \sum_{j=0}^{2\ell_{in}+1} \tag{C.2}$$

$$\sum_{(\ell_{out}ui)} \equiv \sum_{\ell_{out}} \sum_{u=0}^{k(\ell_{out})} \sum_{i=0}^{2\ell_{out}+1} \tag{C.3}$$

$$\sum_{(\ell m)} \equiv \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \sum_{m=0}^{2\ell+1} \tag{C.4}$$

Sets of pairs $(\ell_{out}, |u| = k(\ell_{out}))$ and $(\ell_{in}, |v| = k(\ell_{in}))$ are defined by user per layer. $(\ell_{out}, |u|)$ for one layer is $(\ell_{in}, |v|)$ for the next.

Note. The order of indices in algebraic expressions matches the one in software implementation. However, not all indices are explicit, in a sense that in implementation one physical index may contain several logical indices stacked, due to later having variable length. Padding is not an option due to memory constraints.

C.1 Forward pass

Calculation of the Periodic Convolution comprises two stages:

- Stage 1: calculate $B_{ab(\ell_{out}ui)}$ - equivalent to message function in Ref. [9].
- Stage 2: reduce $B_{ab(\ell_{out}ui)}$ over b - equivalent to message aggregation in Ref. [9].

C.1.1 Routine: Stage 1

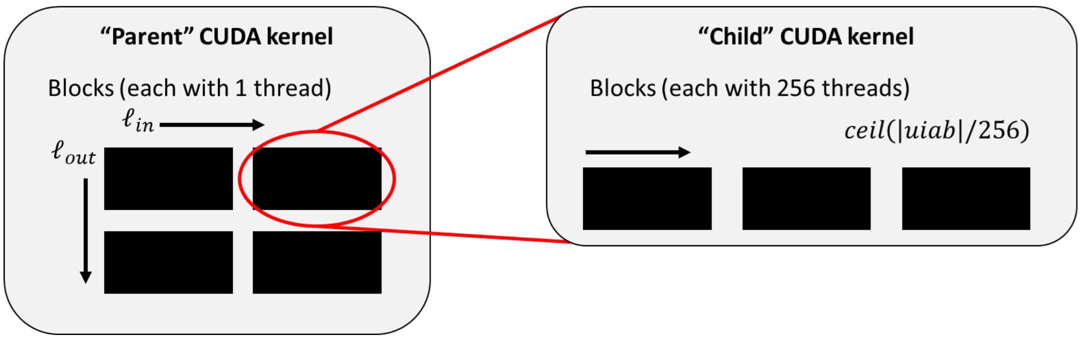

FIGURE C.1: Division of work on CUDA (forward pass)

"Parent" kernel factors in memory offsets corresponding to the ℓ_{out} and ℓ_{in} indices, and then splits the work among "Child" kernels that perform actual calculations. $|uiab|$ means size of the combined set over those indices.

I decided to use dynamic parallelism [50] (Figure C.1) as it simplifies the code while avoiding issues with thread divergence and a huge number of invocations of empty kernels. For example, boundaries of summation over ℓ depend on ℓ_{out}, ℓ_{in}. Having blocks over ℓ_{out}, ℓ_{in} fixes summation boundaries (within each block).

Operations to be used in kernels are computationally cheap: addition and multiplication. At the same time, inputs may be rather large, making interactions with memory a major limiting factor. In order to mitigate this, memory accesses should be coalesced whenever possible. In simple words, it is better if accessed memory locations across threads within a block align. For that reason, it makes sense to transpose some inputs before passing to the kernel.

$$\begin{aligned} R_{(ab)(\ell_{out}\ell_{in}\ell uv)} &\rightarrow R_{(\ell_{out}\ell_{in}\ell uv)(ab)} \\ F_{g(a,b)(\ell_{in}vj)} &\rightarrow F_{(\ell_{in}vj)g(a,b)} \end{aligned}$$

"Parent" kernel factors in memory offsets (denoted with tilde):

$$\begin{aligned} B_{\ell_{out}uiab} &\rightarrow \tilde{B}_{uiab} \\ W_{\ell_{out}\ell_{in}ab} &\rightarrow \tilde{W}_{ab} \\ C_{\ell_{out}\ell_{in}\ell ijm} &\rightarrow \tilde{C}_{\ell ijm} \\ F_{\ell_{in}vjg(a,b)} &\rightarrow \tilde{F}_{vjg(a,b)} \\ R_{\ell_{out}\ell_{in}\ell uvab} &\rightarrow \tilde{R}_{\ell uvab} \end{aligned} \tag{C.5}$$

Note: upper script (n) is dropped, as it is no longer needed to discern input and output features.

"Child" kernel:

$$\tilde{B}_{uiab} += \tilde{W}_{ab} \sum_{v=0}^{k(\ell_{in})} \sum_{j=0}^{2\ell_{in}+1} \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \sum_{m=0}^{2\ell+1} \tilde{C}_{\ell ijm} \tilde{F}_{vjg(a,b)} Y_{\ell mab} \tilde{R}_{\ell uvab} \tag{C.6}$$

where "+=" corresponds to the summation over ℓ_{in} (across blocks), which technically happens within "Child" kernels by means of atomicAdd() function.

Once all kernels completed their tasks, transpose B:

$$B_{(\ell_{out}ui)(ab)} \to B_{(ab)(\ell_{out}ui)}$$

C.1.2 Routine: Stage 2

$$F^{(n+1)}_{a(\ell_{out}ui)} = \frac{1}{\sqrt{|N(a)|}} \sum_{b \in N(a)} B_{ab(\ell_{out}ui)} \tag{C.7}$$

This operation can be done in PyTorch via index_add(). Indices it requires correspond to mapping $h(a,b) \to a$. Keep in mind that mapping is required, because ab is a **single** physical index in implementation.

C.2 Backward pass for the features

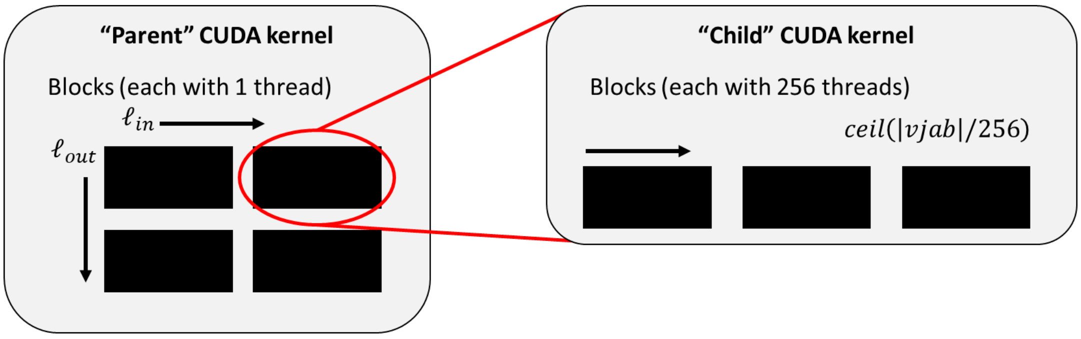

FIGURE C.2: Division of work on CUDA (backward pass for F)

The backward pass for the features is defined as:

$$\nabla F^{(n)}_{b^*(\ell_{in}vj)^*} = \sum_a \sum_{(\ell_{out}ui)} \bar{G}_{a(\ell_{out}ui)} \frac{\partial F^{(n+1)}_{a(\ell_{out}ui)}}{\partial F^{(n)}_{b^*(\ell_{in}vj)^*}} \tag{C.8}$$

where $\bar{G}$ - incoming gradients.

Given Eq. C.1, partial derivative in Eq. C.8 expands as follows:

$$\begin{aligned}\nabla F_{b^*(\ell_{in}vj)^*} = \sum_a \sum_{(\ell_{out}ui)} G_{a(\ell_{out}ui)} \sum_{b \in N(a)} \sum_{\ell_{in}} W_{\ell_{out}\ell_{in}ab} \sum_{(vj)} \sum_{(\ell m)} C_{\ell_{out}\ell_{in}\ell ijm} \times \\ \times Y_{\ell mab} R_{ab\ell_{out}\ell_{in}\ell uv} \delta_{b^* g(a,b)} \delta_{(\ell_{in}vj)^*(\ell_{in}vj)}\end{aligned} \tag{C.9}$$

where $G_{a(\ell_{out}ui)} = \frac{\bar{G}_{a(\ell_{out}ui)}}{\sqrt{|N(a)|}}$ - scaled gradients.

Contraction over Kronecker's delta $\delta_{(\ell_{in}vj)^*(\ell_{in}vj)}$ gives:

$$\nabla F_{b^*(\ell_{in}vj)^*} = \sum_{(ab)} \sum_{(\ell_{out}ui)} \sum_{(\ell m)} W_{\ell_{out}\ell_{in}^* ab} C_{\ell_{out}\ell_{in}^* \ell i j^* m} G_{a(\ell_{out}ui)} Y_{\ell m a b} R_{ab\ell_{out}\ell_{in}^*\ell u v^*} \tag{C.10}$$

$$\sum_{(ab)} \equiv \sum_{a} \sum_{b \in N(a)} \delta_{b^* g(a,b)} \tag{C.11}$$

The calculation of the backward pass for the features comprises two stages based on equations C.10, C.11:

- Stage 1: calculate $\nabla B_{ab(\ell_{in}vj)^*}$
- Stage 2: reduce $\nabla B_{ab(\ell_{in}vj)^*}$ over ab

C.2.1 Routine: Stage 1

For the memory coalescence reasons, inputs should be transposed before passing to the kernel:

$$G_{a(\ell_{out}ui)} \to G_{(\ell_{out}ui)a}$$
$$R_{(ab)(\ell_{out}\ell_{in}^*\ell u v^*)} \to R_{(\ell_{out}\ell_{in}^*\ell u v^*)(ab)}$$

"Parent" kernel factors in memory offsets (denoted with tilde):

$$\begin{aligned} \nabla B_{(\ell_{in}vj)^* ab} &\to \nabla \tilde{B}_{v^* j^* ab} \\ W_{\ell_{out}\ell_{in} ab} &\to \tilde{W}_{ab} \\ C_{\ell_{out}\ell_{in}^* \ell i j^* m} &\to \tilde{C}_{\ell i j^* m} \\ G_{\ell_{out} u i a} &\to \tilde{G}_{uia} \\ R_{\ell_{out}\ell_{in}^*\ell u v^* ab} &\to \tilde{R}_{\ell u v^* ab} \end{aligned} \tag{C.12}$$

"Child" kernel:

$$\nabla \tilde{B}_{v^* j^* ab} \mathrel{+}= \tilde{W}_{ab} \sum_{u=0}^{k(\ell_{out})} \sum_{i=0}^{2\ell_{out}+1} \sum_{\ell=|\ell_{out}-\ell_{in}|}^{\ell_{out}+\ell_{in}} \sum_{m=0}^{2\ell+1} \tilde{C}_{\ell i j^* m} \tilde{G}_{uia} Y_{\ell m a b} \tilde{R}_{\ell u v^* ab} \tag{C.13}$$

where "+=" corresponds to the summation over ℓ_{out} (across blocks), which technically happens within "Child" kernels by means of atomicAdd() function.

Once all the kernels completed their tasks, transpose ∇B:

$$\nabla B_{(\ell_{in}vj)^*(ab)} \to \nabla B_{(ab)(\ell_{in}vj)^*}$$

C.2.2 Routine: Stage 2

$$\nabla F_{b^*(\ell_{in}vj)^*} = \sum_{a} \sum_{b \in N(a)} \delta_{b^* g(a,b)} \nabla B_{ab(\ell_{in}vj)^*} \tag{C.14}$$

This operation can be done in Pytorch via applying index_add() function to $\nabla B_{ab(\ell_{in}vj)^*}$ given $g(a,b)$ mapping as indices.

C.3 Backward pass for the Radial Basis Model outputs

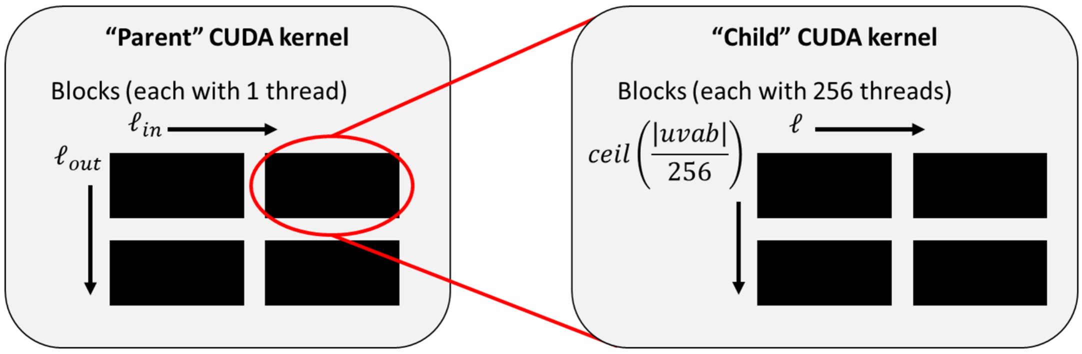
"Parent" CUDA kernel
Blocks (each with 1 thread)
ℓ_{in}
ℓ_{out}
"Child" CUDA kernel
Blocks (each with 256 threads)
$ceil\left(\frac{|uvab|}{256}\right)$
ℓ

FIGURE C.3: Division of work on CUDA (backward pass for R)

The backward pass for the Radial Basis Model outputs is defined as:

$$\nabla R_{(ab)^*(\ell_{out}\ell_{in}f(\ell)uv)^*} = \sum_a \sum_{(\ell_{out}ui)} \bar{G}_{a(\ell_{out}ui)} \frac{\partial F^{(n+1)}_{a(\ell_{out}ui)}}{\partial R_{(ab)^*(\ell_{out}\ell_{in}f(\ell)uv)^*}} \tag{C.15}$$

where $\bar{G}$ - incoming gradients.

Given Eq. C.1, partial derivative in Eq. C.15 expands as:

$$\begin{aligned}\nabla R_{(ab)^*(\ell_{out}\ell_{in}\ell uv)^*} = \sum_a \sum_{(\ell_{out}ui)} G_{a(\ell_{out}ui)} \sum_{b\in N(a)} \sum_{\ell_{in}} W_{\ell_{out}\ell_{in}ab} \sum_{(vj)} \sum_{(\ell m)} C_{\ell_{out}\ell_{in}\ell ijm} \times \\ \times F_{g(a,b)(\ell_{in}vj)} Y_{\ell mab} \delta_{(ab)^*(ab)} \delta_{(\ell_{out}\ell_{in}\ell uv)^*(\ell_{out}\ell_{in}\ell uv)}\end{aligned} \tag{C.16}$$

where $G = \frac{\bar{G}}{\sqrt{|N(a)|}}$ - scaled gradients.

Contraction over both Kronecker's deltas gives:

$$\begin{aligned}\nabla R_{(ab)^*(\ell_{out}\ell_{in}\ell uv)^*} = \sum_{i=0}^{(2\ell^*_{out}+1)} \sum_{j=0}^{(2\ell^*_{in}+1)} \sum_{m=0}^{(2\ell+1)} G_{a^*(\ell^*_{out}u^*i)} W_{\ell^*_{out}\ell^*_{in}a^*b^*} C_{\ell^*_{out}\ell^*_{in}\ell^* ijm} \times \\ \times F_{g(a^*,b^*)(\ell^*_{in}v^*j)} Y_{\ell^* ma^*b^*}\end{aligned} \tag{C.17}$$

Unlike the routines described in the previous sections, this calculation does not benefit from division in stages.

C.3.1 Routine

Before passing to the kernel, inputs should be transposed:

$$F_{g(a^*,b^*)(\ell^*_{in}v^*j)} \to F_{(\ell^*_{in}v^*j)g(a^*,b^*)}$$
$$G_{a^*(\ell^*_{out}u^*i)} \to G_{(\ell^*_{out}u^*i)a^*}$$

"Parent" kernel factors in memory offsets (denoted with tilde):

$$\begin{aligned} \nabla R_{(\ell_{out}\ell_{in}\ell uv)^*(ab)^*} &\to \nabla\tilde{R}_{(\ell uv)^*(ab)^*} \\ G_{(\ell^*_{out}u^*i)a^*} &\to \tilde{G}_{u^*ia^*} \\ W_{\ell^*_{out}\ell^*_{in}a^*b^*} &\to \tilde{W}_{a^*b^*} \\ C_{\ell^*_{out}\ell^*_{in}\ell^*ijm} &\to \tilde{C}_{\ell^*ijm} \\ F_{(\ell^*_{in}v^*j)g(a^*,b^*)} &\to \tilde{F}_{v^*jg(a^*,b^*)} \end{aligned} \tag{C.18}$$

"Child" kernel:

$$\tilde{R}_{(\ell uv)^*(ab)^*} = \tilde{W}_{a^*b^*} \sum_{i=0}^{(2\ell^*_{out}+1)} \sum_{j=0}^{(2\ell^*_{in}+1)} \sum_{m=0}^{(2\ell+1)} \tilde{C}_{\ell^*ijm}\tilde{G}_{u^*ia^*}\tilde{F}_{v^*jg(a^*,b^*)}Y_{\ell^*ma^*b^*} \tag{C.19}$$

Once all the kernels completed their tasks, transpose ∇R:

$$\nabla R_{(\ell_{out}\ell_{in}\ell uv)^*(ab)^*} \to \nabla R_{(ab)^*(\ell_{out}\ell_{in}\ell uv)^*}$$

Appendix D

Real Gaunt coefficients

The complex (real) Gaunt coefficients are coefficients in decomposition of product of two complex (real) spherical harmonics into a combination of single complex (real) spherical harmonics.

$$Y^{\ell_2}_{m_2} Y^{\ell_3}_{m_3} = \sum_{\ell_1=|\ell_2-\ell_3|}^{\ell_2+\ell_3} \sum_{m_1=-\ell_1}^{\ell_1} \tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3} Y^{\ell_1}_{m_1} \tag{D.1}$$

$$Y^{\ell_2}_{m_2} Y^{\ell_3}_{m_3} = \sum_{\ell_1=|\ell_2-\ell_3|}^{\ell_2+\ell_3} \tilde{G}^{\ell_1,\ell_2,\ell_3}_{-(m_2+m_3),m_2,m_3} Y^{\ell_1}_{-(m_2+m_3)} \tag{D.2}$$

Y^{ℓ}_{m} - complex spherical harmonic, $\tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3}$ - complex Gaunt coefficient.

$$Y_{\ell_2 m_2} Y_{\ell_3 m_3} = \sum_{\ell=|\ell_2-\ell_3|}^{\ell_2+\ell_3} \sum_{m_1=-\ell}^{\ell_1} G^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3} Y_{\ell_1 m_1} \tag{D.3}$$

$Y_{\ell m}$ - real spherical harmonic, $G^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3}$ - real Gaunt coefficient.

Note: equation D.2 comes from D.1 based on the fact that complex Gaunt coefficient can be non-zero only if $m_1 + m_2 + m_3 = 0$. This is not true for real Gaunt coefficients, therefore we don't have similar identity for equation D.3.

Properties of real Gaunt (R-Gaunt) coefficients [46]:

1. Invariant to permutations of pairs (ℓ, m), (ℓ_1, m_1), (ℓ_2, m_2)
2. Zero when number of negatives in (m, m_1, m_2) is odd
3. Zero when (ℓ, ℓ_1, ℓ_2) does **not** satisfy $|\ell_1 - \ell_2| \leq \ell \leq \ell_1 + \ell_2$
4. Zero when sum $\ell + \ell_1 + \ell_2$ is odd
5. Zero $|m| > \ell$ for any of (ℓ, m) pairs

Properties #1, #3, #4, #5 are inherited from complex Gaunt coefficients.
Property #1 allows canonical order (I decided to use descending order over m's):

$$m_1 \geq m_2 \geq m_3 \tag{D.4}$$

Explicit expressions for real Gaunt coefficients [47]:
Note: compared to the source, following has different grouping of cases where one of m's is zero. Canonical order $m_1 \geq m_2 \geq m_3$ assumed.

Case A: $\{m_1, m_2, m_3 > 0\}$:

$$G^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3} = \begin{cases} (-1)^{m_1}\frac{\sqrt{2}}{2}\tilde{G}^{\ell_1,\ell_2,\ell_3}_{-m_1,m_2,m_3} & \text{if } m_1 = m_2 + m_3 \\ (-1)^{m_2}\frac{\sqrt{2}}{2}\tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,-m_2,m_3} & \text{if } m_2 = m_1 + m_3 \\ (-1)^{m_3}\frac{\sqrt{2}}{2}\tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,-m_3} & \text{if } m_3 = m_1 + m_2 \\ 0 & \text{otherwise} \end{cases} \tag{D.5}$$

Case B: $\{m_1 > 0; \quad m_2, m_3 < 0\}$:

$$G^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3} = \begin{cases} (-1)^{m_3}\frac{\sqrt{2}}{2}\tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,-m_2,m_3} & \text{if } m_2 = m_1 + m_3 \\ (-1)^{m_2}\frac{\sqrt{2}}{2}\tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,-m_3} & \text{if } m_3 = m_1 + m_2 \\ (-1)^{m_1}\frac{\sqrt{2}}{2}\tilde{G}^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,m_3} & \text{if } m_1 + m_2 + m_3 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{D.6}$$

Case C: $\{m_3 = 0; \quad m_1 = m_2 > 0\}$:

$$G^{\ell_1,\ell_2,\ell_3}_{m_1,m_2,0} = (-1)^{m_1} G^{\ell_1,\ell_2,\ell_3}_{m_1,-m_1,0} \tag{D.7}$$

Case D: $\{m_1 = 0; \quad m_2 = m_3 < 0\}$:

$$G^{\ell_1,\ell_2,\ell_3}_{0,m_2,m_3} = (-1)^{m_2} G^{\ell_1,\ell_2,\ell_3}_{0,m_2,-m_2} \tag{D.8}$$

Due to property #2 all other combinations of signs of m_1, m_2, m_3 yield 0.

Appendix E

Elastic Tensor

E.1 Covariant components

Conversion between Cartesian basis and covariant components is given by:

$$x_\alpha = K_{\alpha i} x_i \tag{E.1}$$

$$x_i = J_{i\alpha} x_\alpha \tag{E.2}$$

where index i signifies Cartesian coordinates, and index α - covariant components.

Matrices K and J are calculated based on following values for x_i, x_α:

$$\begin{bmatrix} x_{i=1} \\ x_{i=2} \\ x_{i=3} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \begin{bmatrix} x_{\alpha=-1} \\ x_{\alpha=0} \\ x_{\alpha=+1} \end{bmatrix} = \begin{bmatrix} Y_{1,-1} \\ Y_{1,0} \\ Y_{1,1} \end{bmatrix} = \begin{bmatrix} -\sqrt{\frac{3}{4\pi}}y \\ -\sqrt{\frac{3}{4\pi}}z \\ -\sqrt{\frac{3}{4\pi}}x \end{bmatrix} \tag{E.3}$$

where Y - real spherical harmonics, as defined by equations 3.2, 3.3.

$$K = -\sqrt{\frac{3}{4\pi}} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad J = -\sqrt{\frac{4\pi}{3}} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \frac{4\pi}{3} K^T \tag{E.4}$$

Note: Mochizuki uses non-conventional definition of complex spherical harmonics that results in matrices K and J being related via complex conjugation, which he denotes as C^* and C respectively [45].

For elastic tensor, transform should be applied to each index:

$$C_{\alpha\beta\gamma\delta} = C_{ijkn} K_{\alpha i} K_{\beta j} K_{\gamma k} K_{\delta n} \tag{E.5}$$

$$C_{ijkn} = C_{\alpha\beta\gamma\delta} J_{i\alpha} J_{j\beta} J_{k\gamma} J_{n\delta} \tag{E.6}$$

E.2 Decomposition equation and reconstruction coefficients

From Ref. [45]:

$$C_{\alpha\beta\gamma\delta} = q_0^m \bigoplus q_2^m \bigoplus \left(s_0^m \bigoplus s_2^m \bigoplus s_4^m\right) \tag{E.7}$$

where C - elastic tensor; q, s - complex spherical harmonics; $m = \alpha + \beta + \gamma + \delta$.

When direct sum $\bigoplus$ replaced with explicit summations, it becomes:

$$C_{\alpha\beta\gamma\delta} = A_m q_0^m + B_m q_2^m + (D_m s_0^m + E_m s_2^m + H_m s_4^m) \tag{E.8}$$

where A, B, D, E, H are fixed coefficients; $m = \alpha + \beta + \gamma + \delta$; $(\alpha, \beta, \gamma, \delta) \in \{-1, 0, +1\}$.

Coefficients are obtained from expanding $(Y_1^\alpha Y_1^\beta)(Y_1^\gamma Y_1^\delta)$ using equation D.2. As noted in Appendix D, there is no correspondence for case of real spherical, therefore we should use more general form as in equation D.3. Consequently, relation between m and $(\alpha, \beta, \gamma, \delta)$ won't be one-to-one, meaning that we would have sum over m's, and making expected expression for real spherical harmonics of the form:

$$C = Ap_{00} + \sum_{m=-2}^{2} B_m p_{2m} + Ds_{00} + \sum_{m=-2}^{2} E_m s_{2m} + \sum_{m=-4}^{4} H_m s_{4m} \tag{E.9}$$

where indices $(\alpha\beta\gamma\delta)$ are implied for C, A, B, D, E, H.

Coefficients A and D stripped from index m, p_{0m} and s_{0m} turned into p_{00} and s_{00}, and bounds of summations set to the respective values, because spherical harmonic component (valid for both real and complex) is zero when $|m| > \ell$.

In order to find coefficients, I expand $(Y_{1\alpha}Y_{1\beta})(Y_{1\gamma}Y_{1\delta})$:

$$Y_{1\alpha}Y_{1\beta} = \sum_{\ell=0}^{2}\sum_{m=-\ell}^{\ell} G_{m,\alpha,\beta}^{\ell,1,1} Y_{\ell m} = G_{0,\alpha,\beta}^{0,1,1} Y_{00} + \sum_{m=-2}^{2} G_{m,\alpha,\beta}^{2,1,1} Y_{2m} \tag{E.10}$$

$$Y_{00}^{(\alpha\beta)} \equiv G_{0,\alpha,\beta}^{0,1,1} Y_{00} \tag{E.11}$$

$$Y_{2m}^{(\alpha\beta)} \equiv \sum_{m=-2}^{2} G_{m,\alpha,\beta}^{2,1,1} Y_{2m} \tag{E.12}$$

$Y_{1\gamma}Y_{1\delta}$ yields the same expressions (up to relabeling). Second equality in E.10 comes from the property #2 in Appendix D. Similarly to the Ref. [45], second step in expansion gives:

- coefficient for q_{00} should stem from interaction $Y_{00}^{(\alpha\beta)}Y_{00}^{(\gamma\delta)}$
- q_{2m}: $Y_{00}^{(\alpha\beta)}Y_{2m}^{(\gamma\delta)} + Y_{2m}^{(\alpha\beta)}Y_{00}^{(\gamma\delta)}$
- (s_{00}, s_{2m}, s_{4m}): $Y_{2m}^{(\alpha\beta)}Y_{2m}^{(\gamma\delta)}$

For q_{2m} it is enough to derive $Y_{00}^{(\alpha\beta)}Y_{2m}^{(\gamma\delta)}$, as $Y_{2m}^{(\alpha\beta)}Y_{00}^{(\gamma\delta)}$ comes from relabeling.

$$Y_{00}^{(\alpha\beta)}Y_{00}^{(\gamma\delta)} = G_{0,\alpha,\beta}^{0,1,1}G_{0,\gamma,\delta}^{0,1,1}Y_{00}Y_{00} = \left(G_{0,\alpha,\beta}^{0,1,1}G_{0,\gamma,\delta}^{0,1,1}G_{0,0,0}^{0,0,0}\right)Y_{00}$$

$$\boxed{A^{(\alpha\beta\gamma\delta)} = G_{0,\alpha,\beta}^{0,1,1}G_{0,\gamma,\delta}^{0,1,1}G_{0,0,0}^{0,0,0}} \tag{E.13}$$

$$\begin{aligned}
Y_{00}^{(\alpha\beta)}Y_{2m_1}^{(\gamma\delta)} &= \sum_{m_1=-2}^{2} G_{0,\alpha,\beta}^{0,1,1}G_{m_1,\gamma,\delta}^{2,1,1}Y_{00}Y_{2m_1} = \\
&= \sum_{m_1=-2}^{2} G_{0,\alpha,\beta}^{0,1,1}G_{m_1,\gamma,\delta}^{2,1,1}\sum_{m=-2}^{2} G_{m,0,m_1}^{2,0,2}Y_{2m} = \\
\left(G_{m,0,m_1}^{2,0,2} = G_{0,m,m_1}^{0,2,2}\right) &= \sum_{m=-2}^{2}\left(\sum_{m_1=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{0,\gamma,\delta}^{0,1,1}G_{0,m,m_1}^{0,2,2}\right)Y_{2m} = \\
\left(G_{0,m,m_1}^{0,2,2} \neq 0 \text{ only if } m_1 = m\right) &= \sum_{m=-2}^{2}\left(G_{m,\alpha,\beta}^{2,1,1}G_{0,\gamma,\delta}^{0,1,1}G_{0,m,m}^{0,2,2}\right)Y_{2m}
\end{aligned}$$

$$\boxed{B_m^{(\alpha\beta\gamma\delta)} = \left(G_{0,\alpha,\beta}^{0,1,1}G_{m,\gamma,\delta}^{2,1,1} + G_{0,\gamma,\delta}^{0,1,1}G_{m,\alpha,\beta}^{2,1,1}\right)G_{m,m,0}^{2,2,0}} \tag{E.14}$$

$$\begin{aligned}
Y_{2m_1}^{(\alpha\beta)}Y_{2m_2}^{(\gamma\delta)} &= \sum_{m_1=-2}^{2}\sum_{m_2=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{m_2,\gamma,\delta}^{2,1,1}Y_{2m_1}Y_{2m_2} = \\
&= \sum_{m_1=-2}^{2}\sum_{m_2=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{m_2,\gamma,\delta}^{2,1,1}\sum_{\ell=0}^{4}\sum_{m=-\ell}^{\ell} G_{m,m_1,m_2}^{\ell,2,2}Y_{\ell m} = \\
\left(G_{m,m_1,m_2}^{\ell,2,2} \neq 0 \text{ only if } \ell \text{ is even}\right) &= \sum_{\ell}^{\{0,2,4\}}\sum_{m=-\ell}^{\ell}\left(\sum_{m_1=-2}^{2}\sum_{m_2=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{m_2,\gamma,\delta}^{2,1,1}G_{m,m_1,m_2}^{\ell,2,2}\right)Y_{\ell m}
\end{aligned}$$

For $\ell = 0 \left(D^{(\alpha\beta\gamma\delta)}\right)$, I use that $G_{0,m_1,m_2}^{0,2,2} \neq 0$ only if $m_2 = m_1$.

$$\boxed{D^{(\alpha\beta\gamma\delta)} = \sum_{m_1=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{m_1,\gamma,\delta}^{2,1,1}G_{0,m_1,m_1}^{0,2,2}} \tag{E.15}$$

$$\boxed{E_m^{(\alpha\beta\gamma\delta)} = \sum_{m_1=-2}^{2}\sum_{m_2=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{m_2,\gamma,\delta}^{2,1,1}G_{m,m_1,m_2}^{2,2,2}} \tag{E.16}$$

$$\boxed{H_m^{(\alpha\beta\gamma\delta)} = \sum_{m_1=-2}^{2}\sum_{m_2=-2}^{2} G_{m_1,\alpha,\beta}^{2,1,1}G_{m_2,\gamma,\delta}^{2,1,1}G_{m,m_1,m_2}^{4,2,2}} \tag{E.17}$$

E.3 Linearly independent subset

Elastic tensor has the same symmetries over indices in covariant components $C_{\alpha\beta\gamma\delta}$ as in Cartesian components C_{ijkn} (see equation 4.3), which follows from **same** transformation being applied separately over each index. This means linearly independent subset can be obtained via selection alone. There are multiple ways to do so (e.g. upper triangular matrix from Voigt notation), here I use increasing order of indices for better clarity.

Symmetries:

$$C_{\alpha\beta\gamma\delta} = C_{\gamma\delta\alpha\beta} = C_{\beta\alpha\gamma\delta} = C_{\alpha\beta\delta\gamma} \tag{E.18}$$

where C - elastic tensor; $\alpha, \beta, \gamma, \delta \in \{-1, 0, +1\}$.

Algorithm 1: Mapping of full set of indices to subset

```
Result: $(\alpha, \beta, \gamma, \delta)$
Given: $\alpha, \beta, \gamma, \delta$
if $\alpha > \gamma$ or $(\alpha = \gamma$ and $\beta > \delta)$ then
    $swap(\alpha, \gamma)$
    $swap(\beta, \delta)$
end
if $\alpha > \beta$ then
    $swap(\alpha, \beta)$
end
if $\gamma > \delta$ then
    $swap(\gamma, \delta)$
end
```

TABLE E.1: Indices of independent entries in elastic tensor C

$'' - '' \equiv -1, '' + '' \equiv +1.$
"id" is a corresponding 0-based index for C flattened into a vector.

$\alpha\beta\gamma\delta$	id
$- - - -$	0
$- - -0$	1
$- - -+$	2
$- - 00$	4
$- - 0+$	5
$- - ++$	8
$+ + ++$	80

$\alpha\beta\gamma\delta$	id
$-0-0$	10
$-0-+$	11
-000	13
$-00+$	14
$-0++$	17
$0+0+$	50
$0+++$	53

$\alpha\beta\gamma\delta$	id
$-+-+$	20
$-+00$	22
$-+0+$	23
$-+++$	26
0000	40
$000+$	41
$00++$	44

Bibliography

[1] *The Nobel Prize in Chemistry 2019*. URL: https://www.nobelprize.org/prizes/chemistry/2019/summary/.

[2] Walter Kohn and Lu Jeu Sham. “Self-consistent equations including exchange and correlation effects”. In: *Physical review* 140.4A (1965), A1133.

[3] DR Bowler and Tsuyoshi Miyazaki. “Methods in electronic structure calculations”. In: *Reports on Progress in Physics* 75.3 (2012), p. 036503.

[4] Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. “Innovation in the pharmaceutical industry: new estimates of R&D costs”. In: *Journal of health economics* 47 (2016), pp. 20–33.

[5] *IBM Watson for Drug Discovery*. URL: https://www.ibm.com/products/watson-drug-discovery. (accessed: 28.09.2019).

[6] *Atomwise: news page*. URL: https://www.atomwise.com/news/. (accessed: 29.09.2019).

[7] Paul Flowers et al. “Chemistry: OpenStax. Chapter 10.6”. In: *DSpace: JSPUI* (2018). URL: https://openlibrary-repo.ecampusontario.ca/jspui/handle/123456789/236.

[8] Lev Landau and Evgeny Lifshitz. *Theory of elasticity: Vol. 7 of course of theoretical physics*. Trans. by J. B. Sykes and W. H. Reid. 2nd ed. Pergamon Press Ltd., 1970.

[9] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1263–1272.

[10] Kristof Schütt et al. “Schnet: A continuous-filter convolutional neural network for modeling quantum interactions”. In: *Advances in neural information processing systems*. 2017, pp. 991–1001.

[11] Tian Xie and Jeffrey C Grossman. “Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties”. In: *Physical review letters* 120.14 (2018), p. 145301.

[12] Nathaniel Thomas et al. “Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds”. In: *arXiv preprint arXiv:1802.08219* (2018).

[13] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. “Clebsch–gordan nets: a fully fourier space spherical convolutional neural network”. In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 10117–10126.

[14] Maurice Weiler et al. “3d steerable cnns: Learning rotationally equivariant features in volumetric data”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 10381–10392.

[15] Chi Chen et al. “Graph networks as a universal machine learning framework for molecules and crystals”. In: *Chemistry of Materials* 31.9 (2019), pp. 3564–3572.

[16] Mario Geiger et al. *github.com/e3nn/e3nn*. Version v0.3-alpha. Mar. 2020. DOI: 10.5281/zenodo.3723557. URL: https://doi.org/10.5281/zenodo.3723557.

[17] Brandon Anderson, Truong Son Hy, and Risi Kondor. "Cormorant: Covariant molecular neural networks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 14510–14519.

[18] Benjamin Kurt Miller. "SE(3) Equivariant Neural Networks for Regression on Molecular Properties: The QM9 Benchmark". MA thesis. Berlin, Germany: Freie Universität Berlin, Mar. 2020. URL: http://dx.doi.org/10.17169/refubium-26900.

[19] Anubhav Jain et al. "The Materials Project: A materials genome approach to accelerating materials innovation". In: *APL Materials* 1.1 (2013), p. 011002. ISSN: 2166532X. DOI: 10.1063/1.4812323.

[20] Shyue Ping Ong et al. "Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis". In: *Computational Materials Science* 68 (Feb. 2013), pp. 314–319. ISSN: 09270256. DOI: 10.1016/j.commatsci.2012.10.028.

[21] *Materials Project: API documentation*. URL: https://wiki.materialsproject.org/The_Materials_API.

[22] G. Kresse and J. Furthmüller. "Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set". In: *Computational Materials Science* 6.1 (1996), pp. 15–50. ISSN: 09270256. DOI: 10.1016/0927-0256(96)00008-0. arXiv: 0927-0256(96)00008 [10.1016].

[23] G. Kresse and J. Hafner. "Ab initio molecular dynamics for liquid metals". In: *Physical Review B* 47.1 (1993), pp. 558–561. ISSN: 01631829. DOI: 10.1103/PhysRevB.47.558. arXiv: 0927-0256(96)00008 [10.1016].

[24] G Kresse and J Furthmüller. "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set". In: *Physical Review B* 54.16 (1996).

[25] Jp Perdew, K Burke, and M Ernzerhof. "Generalized Gradient Approximation Made Simple." In: *Physical review letters* 77.18 (Oct. 1996), pp. 3865–3868. ISSN: 1079-7114. URL: http://www.ncbi.nlm.nih.gov/pubmed/10062328.

[26] S. L. Dudarev et al. "Electron-energy-loss spectra and the structural stability of nickel oxide:An LSDA+U study". In: *Physical Review B* 57.3 (1998), pp. 1505–1509. ISSN: 1098-0121. DOI: 10.1103/PhysRevB.57.1505.

[27] Mariette Hellenbrandt. "The Inorganic Crystal Structure Database (ICSD) — Present and Future". In: *Crystallography Reviews* 10.1 (2004), pp. 17–22. DOI: 10.1080/08893110410001664882.

[28] *ICSD: Main page*. URL: https://icsd.fiz-karlsruhe.de/index.xhtml. (accessed: 05.05.2020).

[29] *Materials Project: Forum*. URL: https://matsci.org/t/theoretical-attribute-is-none/3423.

[30] *Materials Project: Forum*. URL: https://matsci.org/t/misplaced-values-of-hubbard-u/3462.

[31] *Materials Project: Forum*. URL: https://matsci.org/t/ambiguous-run-type/3927.

[32] *MEGNet: Github repository*. URL: https://github.com/materialsvirtuallab/megnet/tree/a47b2acde0901ab7269087bd3ef80730c29955ff.

[33] Matthew D McCluskey and Eugene E Haller. *Dopants and defects in semiconductors*. CRC press, 2018.

[34] Henning Glawe. "Descriptors of Superconductivity". PhD thesis. Berlin, Germany: Freie Universität Berlin, 2018.

[35] C. P. Brock et al., eds. *International Tables for Crystallography*. International Union of Crystallography. DOI: 10.1107/97809553602060000001. URL: https://doi.org/10.1107/97809553602060000001.

[36] *se3cnn: legacy github repository*. URL: https://github.com/mariogeiger/se3cnn. (accessed: 15.05.2020).

[37] Didier Pinchon and Philip E Hoggan. "Rotation matrices for real spherical harmonics: general rotations of atomic orbitals in space-fixed axes". In: *Journal of Physics A: Mathematical and Theoretical* 40.7 (2007), p. 1597.

[38] *Lie learn: Github repository*. URL: https://github.com/AMLab-Amsterdam/lie_learn.

[39] *Wikipedia: Table of spherical harmonics*. URL: https://en.wikipedia.org/w/index.php?title=Table_of_spherical_harmonics&oldid=930589716.

[40] *Wikipedia: Hooke's law for linear spring*. URL: https://en.wikipedia.org/w/index.php?title=Hooke%27s_law&oldid=947255948#For_linear_springs.

[41] Maarten de Jong. *Materials Project Documentation: Elasticity*. URL: https://docs.materialsproject.org/methodology/elasticity/. (accessed: 05.05.2020).

[42] Maarten De Jong et al. "Charting the complete elastic properties of inorganic crystalline compounds". In: *Scientific data* 2.1 (2015), pp. 1–13.

[43] Yakov Itin. "Irreducible matrix resolution of the elasticity tensor for symmetry systems". In: *arXiv preprint arXiv:1812.03367* (2018).

[44] Mildred Dresselhaus, Gene Dresselhaus, and Ado Jorio. "Group theory: application to the physics of condensed matter". In: Springer Science & Business Media, 2007, pp. 456–477.

[45] Eiji Mochizuki. "Spherical harmonic decomposition of an elastic tensor". In: *Geophysical Journal International* 93.3 (1988), pp. 521–526.

[46] Herbert HH Homeier and E Otto Steinborn. "Some properties of the coupling coefficients of real spherical harmonics and their relation to Gaunt coefficients". In: *Journal of Molecular Structure: THEOCHEM* 368 (1996), pp. 31–37.

[47] Didier Pinchon and Philip E Hoggan. "New index functions for storing Gaunt coefficients". In: *International Journal of Quantum Chemistry* 107.12 (2007), pp. 2186–2196.

[48] *Pytorch geometric: Github repository*. URL: https://github.com/rusty1s/pytorch_geometric. (accessed: 21.05.2020).

[49] *DGL: Github repository*. URL: https://github.com/dmlc/dgl. (accessed: 21.05.2020).

[50] *CUDA Toolkit Documentation: Appendix D. CUDA Dynamic Parallelism*. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#cuda-dynamic-parallelism. (accessed: 01.05.2020).