ML-Index Methods

David Moreau

June 24, 2024

My understand of reading through powder diffraction indexing methods papers is that existing algorithms work really well on good data. The SVD-Index index algorithm showed 100% accuracy in their paper with peak lists generated with random unit cells. The biggest issue is the generalization of these algorithms to real experimental data, especially low quality data. So the benchmark for this project is near perfect results on calculated data, or demonstrating superior generality to experimental data. The future development of this project will follow the following development cycle:

1. Demonstrate that this algorithm works well on ideal data by pushing the monoclinic and triclinic indexing rates into the 90's on the current dataset calculated from CCDC / COD entries.
2. Determine appropriate levels of experimental error through characterize experimental peak lists from our smSFX data.
3. Start generalization by regenerating datasets with experimental error and testing the performance. The experimental error will be incorporated in the following order: 1) missing peaks at a realistic rate, 2) error in peak positions, 3) contaminant peaks. Eventually a zero error will need to be included, but is a low priority.

Text in italics is commentary. For example, my hunches, opinions, acknowledgments of errors and bugs in the code, or features and approaches I would like to take, and features that are implemented but not used. Normal case text describes currently implemented methods.

Bravais Lattice	Indexing Rate
aP	61 - 84%
cF	99.5%
cI	100.0%
cP	99.7%
hP	99.5%
hR	92 - 99.4%
mC	81 - 92%
mP	85%
oC	98.5%
oF	99.0%
oI	99.0%
oP	99.5%
tI	99.0%
tP	99.5%

Table 1: Current indexing rates. This is the ability to index a set of 20 diffraction peaks (10 for cubic). If a range is given, the lower number is the rate for finding the exact or symmetry equivalent unit cell. The higher number is the rate for providing a unit cell that can explain the diffraction pattern to within round-off error.

1 Introduction

Powder diffraction indexing is the process of using a 1-dimensional projection of the 3-dimensional diffraction to infer the symmetries that exist within a material. This is a problem as old as crystallography (Runge 1917 *Additional citations - Andrews & Bernstein cite these*) and has been the subject of many analytical efforts. Current trends in machine learning and artificial intelligence have produced new knowledge and tools for data-driven analytics. This project merges recent advancements in analytical methods with knowledge from existing indexing algorithms to produce a new approach to powder diffraction indexing.

This algorithm is developed with general use in mind, but the foremost use case is the small molecule serial crystallography community (Schriber, et. al. 2022). This software is developed as an open-source Python program, in line with the Computational Crystallography Initiative (CITE). Serial femtosecond crystallography data analysis is performed exclusively on super-computers and computational clusters. ML-Index is developed specifically for mass-distributed computing.

1.1 Inspirational Quotes

- de Wolff 1957: The 'indexing problem' is essentially a puzzle: it cannot be stated in rigorous terms, or be solved in a general way by clear-cut methods, because of the unpredictable distribution of unobserved lines in a powder pattern. It would be quite an easy puzzle if errors of measurement did not exist. This added inconvenience, however, is enough to raise some doubt as to whether the determination of the unit cell of an arbitrary polycrystalline phase is possible in, say, an average of half a day's work or less.

  Le Bail (2004) provides some quotes:

- 'Indexing is more an art than a science,'
- 'It is entirely the users responsibility to decide whether any of the suggested unit cells is the correct cell' (ITO manual)
- 'Powder indexing works beautifully on good data, but with poor data it will usually not work at all' (Shirley, 1980)
- 'DICVOL proposes solutions, the user disposes of them' (DICVOL manual)
- Users now want solutions fast, without thinking too much, ignoring that 'part of the beauty of structure determination by powder diffraction does consist in its complexity, i.e., in the lack of complete automatism as well as in the necessity of a careful and sagacious human interpretation of the experimental data,' (anonymous reviewer)

1.2 Powder diffraction indexing

Diffraction is produced in 3-dimensional spherical coordinates - reciprocal space. In the powder diffraction method, the 3d diffraction pattern is recorded in the radial dimension, the distance of the measured diffraction from the center of the spherical coordinate system. The recorded pattern is a series of peaks with the location of the peaks representing the spacing of Miller planes through the crystal's unit cell. In the indexing problem, the objective is to determine the unit cell level symmetries of the crystal that produced this pattern. When the diffraction is in coordinates of $q = \frac{2\sin(\theta)}{\lambda}$, where θ is the diffraction angle and λ is the wavelength, the crystal's unit cell can be related to the observed peak locations with through the general equation:

$$q_{hkl}^2 = \frac{1}{V}\left(S_{hh}h^2 + S_{kk}k^2 + S_{ll}l^2 + S_{hk}hk + S_{hl}hl + S_{kl}kl\right), \tag{1}$$

$S_{hh} = b^2c^2\sin^2(\alpha)$,
$S_{kk} = a^2c^2\sin^2(\beta)$,
$S_{ll} = a^2b^2\sin^2(\gamma)$,
$S_{hk} = abc^2\left[\cos(\alpha)\cos(\beta) - \cos(\gamma)\right]$,
$S_{hl} = a^2bc\left[\cos(\beta)\cos(\gamma) - \cos(\alpha)\right]$,
$S_{kl} = ab^2c\left[\cos(\alpha)\cos(\gamma) - \cos(\beta)\right]$,
$V(\text{Volume}) = abc\sqrt{1 - \cos^2(\alpha) - \cos^2(\beta) - \cos^2(\gamma) + 2\cos(\alpha)\cos(\beta)\cos(\gamma)}$.

This general equation assumes no constraints among the crystal lattice lengths, a, b, c, and angles, α, β, γ. It is simplified by the lattice parameters constraints defining the seven lattice systems,

triclinic : $a \neq b \neq c,\ \alpha \neq \beta \neq \gamma \neq 90^o$

cubic : $a = b = c,\ \alpha = \beta = \gamma = 90^o$

$$q_{hkl}^2 = \frac{h^2 + k^2 + l^2}{a^2}, \tag{2}$$

tetragonal : $a = b \neq c,\ \alpha = \beta = \gamma = 90^o$

$$q_{hkl}^2 = \frac{h^2 + k^2}{a^2} + \frac{l^2}{c^2} \tag{3}$$

hexagonal : $a = b \neq c,\ \alpha = \beta = 90^o\ \gamma = 120^o$

$$q_{hkl}^2 = \frac{4}{3}\frac{h^2 + hk + k^2}{a^2} + \frac{l^2}{c^2} \tag{4}$$

rhombohedral : $a = b = c,\ \alpha = \beta = \gamma \neq 90^o$

$$q_{hkl}^2 = \frac{(h^2 + k^2 + l^2)\sin^2(\alpha) + 2(hk + kl + hl)(\cos^2(\alpha) - \cos(\alpha))}{a^2(1 - 3\cos^2(\alpha) + 2\cos^3(\alpha))} \tag{5}$$

orthorhombic : $a \neq b \neq c,\ \alpha = \beta = \gamma = 90^o$

$$q_{hkl}^2 = \frac{h^2}{a^2} + \frac{k^2}{b^2} + \frac{l^2}{c^2} \tag{6}$$

monoclinic : $a \neq b \neq c,\ \alpha = \gamma = 90^o, \beta \neq 90$

$$q_{hkl}^2 = \frac{h^2}{a^2\sin^2(\beta)} + \frac{k^2}{b^2} + \frac{l^2}{c^2\sin^2(\beta)} - \frac{2hl\cos(\beta)}{ac\sin^2(\beta)} \tag{7}$$

The triclinic lattice system imposes no unit cell parameter constraints and is known as the triclinic lattice system. The general case gives the peak spacing formula.

These equations can be rewritten in the reciprocal space unit cell parameters, $a*$, $b*$, $c*$, $\alpha*$, $\beta*$, $\gamma*$, which are related to the real-space unit cell parameters through the inverse of the metric tensor,

$$S = \begin{pmatrix} a^2 & ab\cos(\gamma) & ac\cos(\beta) \\ ab\cos(\gamma) & b^2 & bc\cos(\alpha) \\ ac\cos(\beta) & bc\cos(\alpha) & c^2 \end{pmatrix} = \begin{pmatrix} a^{*2} & a^*b^*\cos(\gamma^*) & a^*c^*\cos(\beta^*) \\ a^*b^*\cos(\gamma^*) & b^{*2} & b^*c^*\cos(\alpha^*) \\ a^*c^*\cos(\beta^*) & b^*c^*\cos(\alpha^*) & c^{*2} \end{pmatrix}^{-1} \tag{8}$$

Eq. 1-7 can be rewritten using reciprocal space unit cells,
triclinic : $a^* \neq b^* \neq c^*,\ \alpha^* \neq \beta^* \neq \gamma^* \neq 90$

$$q_{hkl}^2 = a^{*2}h^2 + b^{*2}k^2 + c^{*2}l^2 + 2a^*b^*\cos(\gamma^*)hk + 2a^*c^*\cos(\beta^*)hl + 2b^*c^*\cos(\alpha^*)kl \tag{9}$$

cubic : $a^* = b^* = c^*,\ \alpha^* = \beta^* = \gamma^* = 90^o$

$$q_{hkl}^2 = \left(h^2 + k^2 + l^2\right)a^{*2} \tag{10}$$

tetragonal : $a^* = b^* \neq c^*,\ \alpha^* = \beta^* = \gamma^* = 90^o$

$$q_{hkl}^2 = \left(h^2 + k^2\right)a^{*2} + l^2c^{*2} \tag{11}$$

hexagonal : $a^* = b^* \neq c^*,\ \alpha^* = \beta^* = 90^o\ \gamma^* = 60^o$

$$q_{hkl}^2 = \left(h^2 + hk + k^2\right)a^{*2} + l^2c^{*2} \tag{12}$$

rhombohedral : $a^* = b^* = c^*,\ \alpha^* = \beta^* = \gamma^* \neq 90^o$

$$q_{hkl}^2 = (h^2 + k^2 + l^2)a^{*2} + 2(hk + kl + hl)a^{*2}\cos(\alpha^*) \tag{13}$$

orthorhombic : $a^* \neq b^* \neq c^*,\ \alpha^* = \beta^* = \gamma^* = 90^o$

$$q_{hkl}^2 = h^2a^{*2} + k^2b^{*2} + l^2c^{*2} \tag{14}$$

monoclinic : $a^* \neq b^* \neq c^*,\ \alpha^* = \gamma^* = 90^o, \beta^* \neq 90$

$$q_{hkl}^2 = h^2a^{*2} + k^2b^{*2} + l^2c^{*2} + 2a^*c^*\cos(\beta^*)hl \tag{15}$$

Eq. 8-15 can be rewritten as a linear function of reciprocal unit cell parameters and Miller indices,

$$q_{hkl}^2 = x_{hh}h^2 + x_{kk}k^2 + x_{ll}l^2 + x_{hk}hk + x_{hl}hl + x_{kl}kl \tag{16}$$

$x_{hh} = a^{*2}$,
$x_{kk} = b^{*2}$,
$x_{ll} = c^{*2}$,
$x_{hk} = 2a^*b^*\cos(\gamma^*)$,
$x_{hl} = 2a^*c^*\cos(\beta^*)$,
$x_{kl} = 2b^*c^*\cos(\alpha^*)$.

$$\vec{q}_{hkl}^2 = H \cdot \vec{x}_{nn}. \tag{17}$$

In the linear model, Eq. 17, the observed peak positions are within the vector $\vec{q}_{hkl}^2$, information about the Miller indices is contained in matrix H, and information about the unit cell is contained in vector $\vec{x}_{nn}$. The implication of this linear model is that given a set of observed peak positions and a hypothesis for Miller indices or unit cell, linear methods can be employed to analytically determine the complementary information.

1.2.1 Difficulties with perfect data

In practice, obtaining the solution to the inverse problem of Eq. 1-7 is hindered by numerous obstacles which can be separated into two types; those that will occur with "perfect data" and those that occur due to measurement errors.

Exisiting powder diffraction indexing methods generally perform well on high quality data and the primary suggestion for increasing the odds of a solution is to obtain better data (Bergmann 2004, Shirley 2003). In the case of perfect data, such as powder diffraction patterns calculated from known crystal structures, existing algorithms should perform extremeley well. Coehlo (2004) evaluated the SVD-Index on randomly generated triclinic unit cells and consistently recover the correct triclinic answer. How well existing algorithms can index unit cells calculated from known structures has not been established, it is likely very high.

For cubic lattice systems, the single free lattice parameter, a can be found rather simply by pencil and paper, each peak occurs at an integer value divided by a^2. For tetragonal, hexagonal, and rhombohedral with two free lattice parameters, solutions can be reliably found without computation aids through graphical methods (Hull & Davey, 1921). The difficulty begins with orthorhombic cases of three free lattice parameters, however, computational methods reliably identify the correct unit cell (CITE). Monoclinic and triclinic lattice systems are known to be difficult for computational algorithms.

Prediction of unit cells from perfect data is hindered by amibiguities in the unit cell representation. The "unit cell" in these relations is an arbitrary mathematical representation of the repeating unit within a crystal as defined by convention. The IUCR defines the conventional unit cell as the smallest volume unit cell with a right-hand basis and edges along the lattice's symmetry directions. This is ambiguous, and there are multiple "settings" that describe the same unit cell. For example, the orthorhombic unit cell can be placed with any permutation of the unit cell lattices. In the monoclinic lattice system, convention dictates that the unit cell be so the monoclinic angle is β and it is obtuse. The monoclinic unit cell can be defined just as well with the monoclinic angle at α or γ and as an acute angle. Additionaly, there are transformations beyond these simple permutations that convert C (base) and I (body) centered lattices or to move a glide plane from along a lattice vector to a diagonal.

There are additional ambiguities when considering the centered Bravais lattices which exist for all lattice systems except triclinic. In these cases, the minimum volume unit cell that describes the crystal's repetition is a triclinic (OR XXX???) lattice system. However, a larger volume unit cell also describes the crystal at a higher degree of symmetry, lending easier mathematical analysis. Convention dictates the use of the centered Bravais lattices

There are other definitions that seek to remove this ambiguity, such as the reduced unit cell (Buerger 1957, CITE SELLING REDUCTION), or the $s6$ (CITE) and $g6$ spaces (Andrews & Bernstein 1988). These definitions are still retain ambiguities (Andrews 2019).

Prediction of unit cells from perfect data is hindered by uniqueness. Unit cells are not generally unique. There are special cases where multiple, conventional unit cells can produce the same diffraction peak spacings. These unique cells are not simply different settings or representations of the same unit cell, but completely different unit cells. These cases have been tabulated, allowing their resolution (Oishi-Tomiyasu, 2016).

The inverse problem posed by Eq. 1-7 is underdetermined, there are more unknowns than knowns. The powder diffraction indexing problem usually starts with a set of 20 observed peak positions $\{q_{obs}\}_{hkl}$. The lattice system equations relate each peak in the set to the same six lattice parameters. The relationships in Eq. 1 - 7 are relatively simple, and as shown in Eq. 1, can be formulated as a set of linear equations. However, each of the 20 equations is parameterized by a unique set of Miller indices $\{hkl\}$. If n_k peaks are chosen, there will be between $3n_k + 1$ and $3n_k + 6$ unknown parameters, there will always be three times more unknown parameters than equations. This is an under-determined inverse problem - more parameters being predicted than constraints that can be imposed. Determining the unknown parameters h, k, & l has long been recognized as the primary obstacle to powder diffraction indexing (de Wolff, 1961; de Wolff, 1968).

Eq. 1-7 shows the parameters being predicted are a mix of continuous and discrete values. The unit cell parameters are continuous values, however the space they exist in is not a simple cartesian space (Andrews, 2019). The discreteness of the Miller indices adds additional challenges. Most advanced numerical methods struggle with discrete values. Further, our analysis treats Miller indices as sets of three integers, making then categorically discrete values with nominal labels without an easily defined relations between them.

1.3 Overview of existing powder diffraction indexing algorithms

Shirley (2003) gives a good overview of the history of powder diffraction indexing.

1.3.1 Dichotomy method

ADD DISCUSSION ON DICVOL.

1.3.2 Zone Indexing

The zone indexing method, first described by Runge (1917), rediscovered by Ito (1949), refined by de Wolff (1957, 1958), and implemented computationally by Visser (1969) demonstrates a methodological approach to determining the angles of a triclinic lattice. For the monoclinic case, the method starts with the assumption that the reciprocal lattice vectors a^* and ca^* have been determined by the position of the low angle reflections. If the monoclinic angle, β^* is 90^o, there will be a diffraction peak at $q^2_{h0l} = h^2a^{*2} + l^2c^{*2}$. When the monoclinic angle is not 90^o, this peak will be split, with two peaks occurring at $q^2_{h0l} = h^2a^{*2} + l^2c^{*2} + 2hla^*c^* \cos(\beta^*)$ and $q^2_{h0-l} = h^2a^{*2} + l^2c^{*2} - 2hla^*c^* \cos(\beta^*)$. The monoclinic angle is obtained from difference of these two peaks, $\cos(\beta^*) = \frac{q^2_{h0l} - q^2_{h0-l}}{4hla^*c^*}$.

This method demonstrates the difficulty in determining lattice angles from powder diffraction. First, an assumption is made that the lattice parameter lengths have been accurately determined. Then, a both sides of the split peak must be observed and identified. Lastly, the angle is determined from a combination of two peak positions and two lattice parameters. In practice, this is performed through a guess and check method and requires powder diffraction patterns determined at high accuracy.

This method should struggle with poor data. Missing low angle peaks will preclude lattice parameter determination. Contaminant peaks will add confusion to lattice parameter determination and identification of peak pairs. Error in peak positions will compound to a larger error in angle.

Machine learning methods for unit cell determination have not been demonstrated to determine unit cell lengths to the precision necessary to use in the zone indexing algorithm.

1.3.3 SVD-Index

The method presented here draws inspiration heavily from the SVD-Index indexing presented by Coehlo (2003). This algorithm takes a generate-and-test approach to the problem, it starts with random guesses of the lattice parameters, then iteratively assigns Miller indices and optimizes the lattice parameters to explain the observed diffraction spacing's. This is a complex algorithm and an abridged overview is given here to support the development of ML-Index. SVD-Index consists of three primary components: 1) Initial unit cell generation; 2) Miller index assignment; and 3) Unit cell optimization.

SVD-Index generates initial unit cell candidates by randomly generating numbers for the unit cell parameters, then rescaling the parameters to match a given unit cell volume. The algorithm incrementally increases the unit cell volume that is used for rescaling until a solution is found. This is a brute-force grid search of the entire parameter space. While inefficient, SVD-Index compensates with speed. Candidate entries are rapidly tested, with nearly 100,000 candidates tested in a matter of seconds. This is a very powerful approach, no assumptions are made regarding how the input diffraction maps onto unit cell parameters. These assumptions could easily lead to algorithm omitting the neighborhood of the true unit cell parameters from the search space in the presence of contaminate diffraction or heavily missing peaks, causing it to fail to find the correct unit cell parameters.

Miller index assignment starts by establishing a reference set of Miller indices. This is done by calculating all possible Miller indices up to a certain resolution cutoff given by the unit cell volume. For each reference Miller index and candidate unit cell, the forward models (Eq. 1-7), are used to calculate reference peak spacings, $\{q_{ref}\}_{hkl}$. Observed peaks are assigned the Miller index associated with the closest reference peak.

Unit cell parameters are optimized to match the agreement between the observed, $\{q_{obs}\}_{hkl}$, and calculated $\{q_{cal}\}_{hkl}$ peak spacings. This is performed using weighted least squares solved by singular value decomposition (SVD). The forward models (Eq. 1-7) are formulated as a linear function of Miller indices, H, and a transformation of the unit cell parameters, $\vec{x}$, $\vec{q}_{cal} = H\vec{x}$. The least squares solution to $\vec{x}$ to $w_{hkl}\vec{q}_{obs} = w_{hkl}H\vec{x}$, where w is a set of weights for each peak, is then found using SVD (CITE). The weight function is

$$w_{hkl} = d^m_{obs,hkl}|2\theta_{obs,hkl} - 2\theta_{cal,hkl}|I_{obs}. \tag{18}$$

Here, $d_{obs,hkl}$ and $2\theta_{obs,hkl}$ are the observed peak spacings in resolution (d-spacing: $d = 1/q$) and angular units. $2\theta_{cal,hkl}$ is the calculated peak spacing in angular units. I_{obs} are the peak's integrated intensity. The ideal exponent, m, was noted to be 4, however using a random number generated between 0 and 4 was found to be the superior choice. This randomness is important to note, it is suggestive that a greedy algorithm, one that always picks the most probable next step, will not be able to find the correct unit cells.

This algorithm works by assuming a lattice system, and exhaustively searching for a solution within each lattice system.

1.4 Data-driven indexing approach

ML-Index seeks to improve the three primary components of SVD-Index using machine learning and Monte-Carlo inspired randomization.

Machine learning models are used to create random unit cell generators to reduce the initial search space. These are either derived by random sampling from neural network predictions of unit cell distributions, or individual trees in random forest models. For each Bravais lattice, an ensemble of these models is trained on data grouped by unresolvable unit cell ambiguities or expected differences in peak spacing "distributions" caused by different patterns of systematic absences.

A separate unit cell generation model is developed based on Miller index templates, in a manner similar to TREOR (Make sure this is a good comparison and cite), however done at a much larger scale. For each Bravais lattice, thousands of random sets of Miller indices are generated based on the observed distribution of Miller indices in a training set. Given a set of observed peak spacings, unit cells can be determined for each template using least-squares methods.

Thousands of unit cell candidates are randomly generated from these models and are optimized by iteratively assigning Miller indices and determining new unit cells. Key to this optimization is randomization. We choose to apply randomization to the Miller index assignment process. A difficulty in this randomization is the categorically discrete nature of the Miller indices. We address this challenge by determining a distribution over Miller index assignments, conditional on a given unit cell. This converts the categorically discrete variables to a continuous variable - a probability distribution - that is easier apply randomization to. This is performed by creating an array of pairwise differences between the observed peak positions and those calculated from the reference Miller indices, $q^2_{obs,hkl} - q^2_{ref,hkl}$. This array is used as input to a neural network classifier.

2 Methods

2.1 Data

Crystallographic data for training the machine learning models are sourced from the Cambridge Structural Database (CSD) and the Crystallographic Open Database (COD). These databases do not contain experimental data, they contain structural information about materials that were determined experimentally. Powder diffraction patterns are calculated from entries in these databases.

True experimental data, as in experimentally measured powder diffraction patterns are sourced from the RRUFF database (Lafuente, et. al., 2015).

I have downloaded the RRUFF database and done some initial exploration.It is very clear that there are serious curation issues that need to be resolved to use this database.

2.1.1 Preparation

The first step of the dataset generation process is an initial quality control of entries in the CSD and COD. The following criteria must be met by an entry for its inclusion in either training or validating the machine learning models.

1. Chemical formula must be listed.
2. Spacegroup number must be listed and between 1 and 230, inclusively.
3. The Hermann–Mauguin symbol must be obtainable from the listed spacegroup symbol.
4. The crystal system must be listed and compatible with the given conventional unit cell parameters. Crystal systems are the same as lattice systems presented in Eq. 1-7, except with hexagonal and rhombohedral grouped.
5. A Niggli reduction, conversion from the conventional to reduced setting, must be successful.
6. Unit cell volume calculated from the given conventional and reduced unit cell parameters must match the given volume.
7. Unit cell volume must be consistent with the number of atoms in the unit cell deduced from the chemical formula. The given volume in $\AA^3$ must be within an order of magnitude of 18 times the number of atoms in the unit cell (CITE).

The next step is a duplicate removal step performed on the CSD and COD independently. A duplicate is defined as the following:

1. Belongs to the same Bravais lattice and,
2. The unit cell has the same chemical composition deduced from the chemical formula & the unit cell volume is within 5%.

3. or, The unit cell volume is exactly the same.

In the case of duplicate entries, the entry with the lowest reported r-factor is chosen. Databases are merged by first taking all the unique entries from the CSD, then only adding entries from the COD if they are not duplicated with an existing CSD entry. For the chemical composition of the unit cell, hydrogen and deuterium atoms, and atoms that comprise less than 5% of the total number of atoms in the unit cell are not taken into consideration. Many studies test the effects of changing one atom in a material. These entries could have nearly the same unit cell and very similar diffraction patterns. Omitting atoms that make up a small portion of the total number of atoms in the unit cell should consider these entries as duplicates.

I get the best performance for unit cell prediction when I use wide shallow neural networks, this was also observed by Vanessa. My understanding of wide shallow neural networks is they tend to work more by memorization as opposed to generalization. A strict duplication removal procedure could prevent the models memorizing identical patterns as opposed to generalization.

Bravais Lattice	Total entries CSD	Unique entries CSD	Total entries COD	Unique entries COD	COD entries not in CSD
aP	317,251	164,254	121,422	76,449	7,284
cF	1,171	870	9,236	3,343	3,121
cI	1,477	1,005	2,783	1,505	1,218
cP	2,859	2,074	4,437	2,687	2,090
hP	13,839	9,961	13,912	9,549	6,208
hR	12,749	8,601	8,768	5,892	3,192
mC	127,726	79,610	54,344	39,016	7,135
mP	493,166	207,248	183,224	97,372	9,425
oC	8,998	7,080	6,131	4,761	2,390
oF	4,658	3,708	2,021	1,718	386
oI	2,587	2,016	2,094	1,623	934
oP	186,790	87,848	72,705	43,483	7,854
tI	9,856	7,503	8,244	6,030	3,488
tP	15,388	11,204	9,841	7,010	3,112

Table 2: Duplicate entries in the CSD & CSD

Datasets are generated from the unique entries in both the CSD and COD using the CCDI API. The following information is extracted from each entry:

- Diffraction pattern
- Database source
- Identifier (If from CSD)
- Local cif file name (If from COD)
- Spacegroup number
- Bravais Lattice
- Lattice system
- Hermann–Mauguin spacegroup symbol
- Unit cell parameters
- Unit cell volume
- Peak lists

The diffraction pattern is calculated between $2\theta = 0^o - > 60^o$ in steps of 0.02^o with peak breadths of 0.1^o full width at half max. It is calculated assuming the Cu K-α wavelength of 1.54 Å. It is used to determine the peak list and is retained, although not used after dataset generation.

Two peak lists are included, these are 1) the first 60 non-systematically absent peaks and 2) the first 60 peaks that could be realistically observed in a diffraction pattern. For a diffraction peak to be included in peak list 2, each of the following conditions must be met:

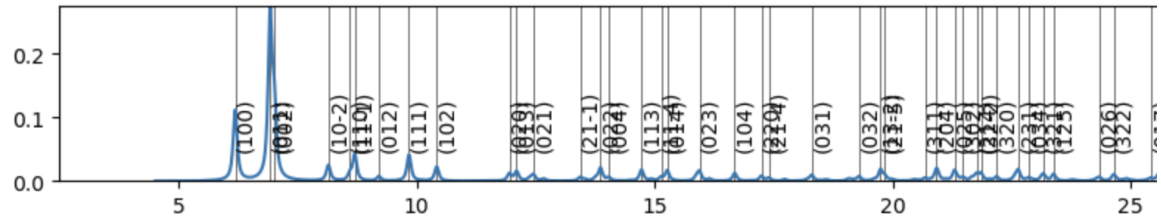
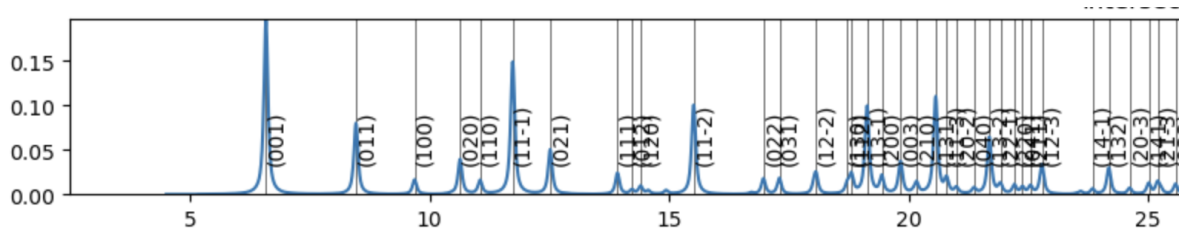


Figure 1: Randomly selected example of the peak list generation. The blue curve is the powder diffraction pattern and the vertical lines are the peaks that were retained.



1. Not systematically absent.
2. The diffraction pattern is normalized such that $\Sigma\ I^2 = 1$. The intensity at the peak position of the normalized diffraction pattern must be larger than 0.005.
3. The second derivative of the diffraction pattern at the peak position must be less than -1.
4. The peak must not be within 0.1^o / 1.5 of another peak. If so, the peak with the largest intensity is retained.

The 0.1 FWHM figure is arbitrary. My impression is that a good diffraction pattern will be narrower. de Wolff 1957 provides 0.05 (FWHM) as their instrumental broadening resolution. The 1.5 was chosen by adjusting to a point where peaks were discarded when they were too close I feel like there should be a simple rule of thumb for this...

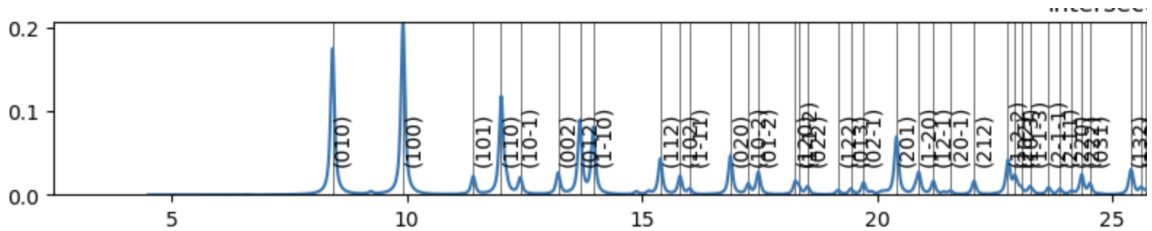
Two representations of the unit cell are included. First is the unit cell is taken directly from the the database. Second is a reindexed unit cell. Each entry is reindexed to a common setting for all entries with the same Bravais lattice, in an attempt to reduce ambiguities.

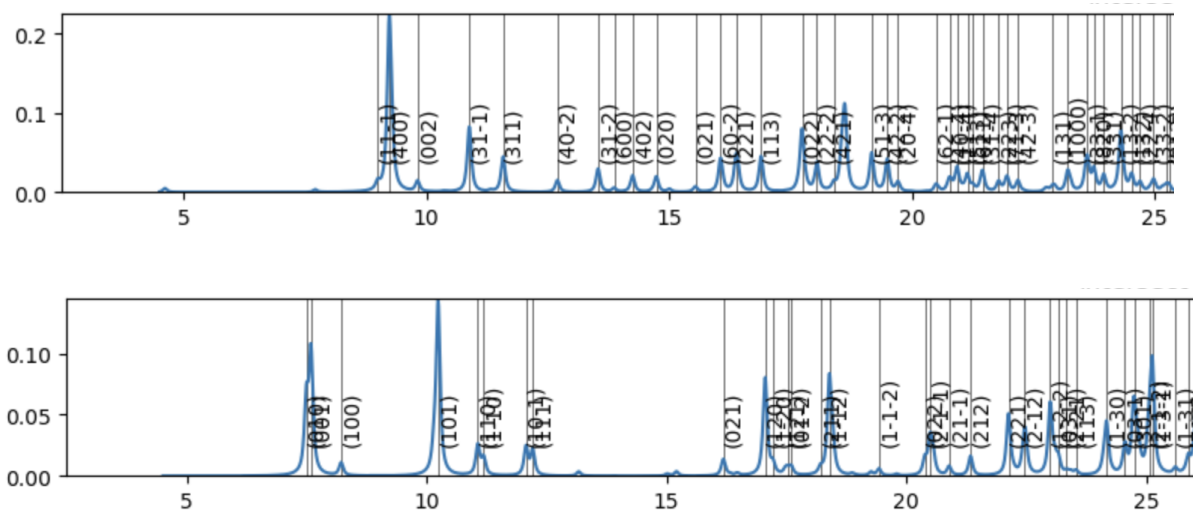
For the cubic, tetragonal, and hexagonal lattice systems, no reindexing is performed because there is no ambiguities in the unit cell representations.

For the rhombohedral lattice system, unit cells can be either placed in either a rhobohedral or hexagonal setting and frequently are listed in databases in the hexagonal settings. All entries are reindexed the rhombohedral setting.

For the orthorhombic lattice system, primitive (oP), face-centered (oF), and body-centered (oI) entries are reindexed so the unit cell lengths, a, b, c are in increasing lengths. Base-centered (oC) entries are placed in the C-centered (as opposed to A or B centered) setting such that cell length a ¡ b. The base-centered entries could be placed in a setting such that unit cell lengths increase, however no common set of systematic absences will exist for these entries. Placing these entries all in the c-centered setting ensures the $h + k = 2n$ reflection condition must be met for all entries.

For monoclinic lattices, centered entries (mC) are conventionally placed in the C-centered setting. For this work centered monoclinic entries are all placed in an equivalent body-centered setting. This allows further





reindexing to enforce the lattice length a is shorter than c. For this reason as well, primitive entries (mP) with a glide plane, *P 1 a 1* for example, are reindexed so the glide plane is along the diagonal, *P 1 n 1*.

Triclinic lattices are reindexed by applying a Selling reduction (Andrews 2019) and then choosing a reflection with increasing lattice lengths ($a < b < c$). The resulting unit cells have all obtuse angles.

2.1.2 Grouping

Ensembles of machine learning models are created by training multiple models on different groupings of data. These groupings are performed according to the hierarchy:

1. Lattice system, then
2. Bravais lattice, then
3. Patterns in reflection conditions.

Grouping hierarchies 1 and 2 are well defined, and hierarchy 3 is subjective. This hierarchy is roughly based on the powder extinction class. In the original paper proposing using artificial neural networks to predict unit cell parameters (Habershon 2004), data was grouped into different powder extinction classes, and models were trained specifically for these groups.

To be clear what is meant by patterns in reflection conditions, consider the primitive orthorhombic space groups P 2 2 2 and P 21 21 21. P 2 2 2 has no systematic absences. The P 21 21 21 spacegroup has screw axis symmetry along the a, b, and c axes, leading to the reflection conditions that Miller index h must be even when $k = l = 0$; the $h00$ reflection must have an even h. The same condition is imposed on k and l for $0k0$ and $00l$ respectively. In the case that a mathematical model that determines the orthorhombic unit cell parameters, and does so by placing all of its emphasis on the low angle peaks which are more likely to have Miller indices 100, 010, and 001, It would first need to identify the difference between the P 2 2 2 and P 21 21 21 entries before predicting the unit cell parameters. If it cannot distinguish these cases, it should have great difficulty in predicting the unit cell parameters.

It should be noted that Hierarchy 3 splits up entries with the same space group. Consider spacegroup 18 which has two screw axes and one axis without a symmetry element. This one spacegroup contains spacegroup symbols P 21 21 2, P 21 2 21, and P 2 21 21. The P 21 21 2 spacegroup will have two reflection conditions $h00; h = 2n$ and $0k0; k = 2n$, and no reflection condition for l. The other two space symbols represent different permutations of the axes and reflection conditions. Again, orthorhombic crystals are reindexed so $a < b < c$, so all three settings will be contained in the training/testing datasets. Spacegroup 18 is split into three different groups, where they will be grouped with entries from other spacegroups with similar reflection conditions. On the other hand, consider spacegroup 17, with one screw axis and two axes without a symmetry element. This spacegroup contains symbols P 21 2 2, P 2 21 2, and P 2 2 21. Ideally, they would be split into different groups like Spacegroup 18, however, spacegroup 17, and other spacegroups with similar patterns in reflection conditions, have very few entries, so they are grouped into a single group due to limited data.

Specifications for the different groups can be found at "data/GroupSpec_*lattice_system*.xlsx" in the Github repository.

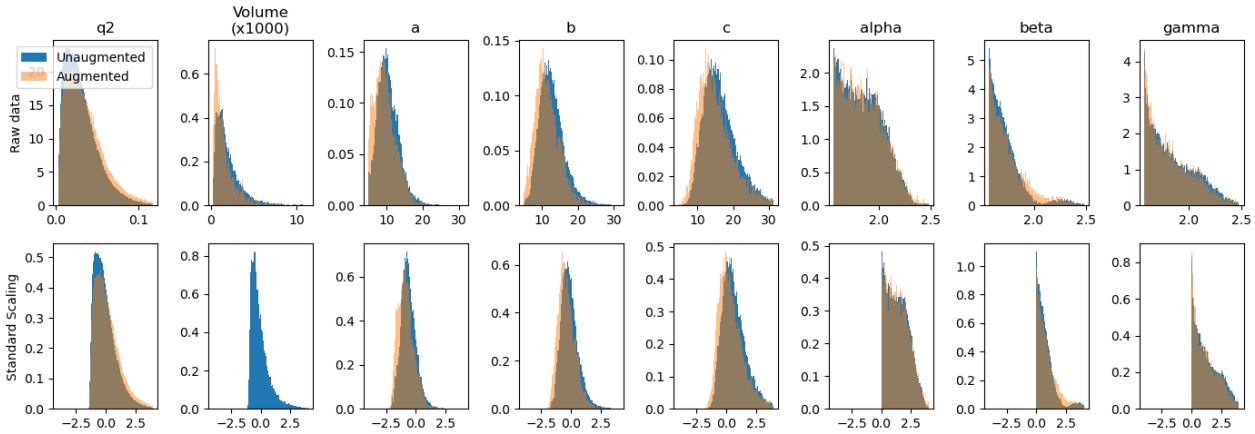


Figure 2: Data distributions for the triclinic lattice system. Top row plots the raw parameters for the unaugmented data in blue and augmented data in orange. The bottom row plots the parameters after scaling. The left column are the input peak spacings. The following columns are the unit cell parameters

2.1.3 Training / Validation split

An 80 / 20% split is made for the training and validation sets. This split occurs at the level of the the Hermann–Mauguin spacegroup symbol. So the separation of training and validation data is performed in groups of common Hermann–Mauguin spacegroup symbol.

2.1.4 Parameter scaling

Neural networks perform best with inputs and outputs that are on the scale of one. So there is typically a rescaling of inputs and outputs so they look like a Normal distribution, or fit between 0 and 1. Training occurs considering each lattice system at a time. For a given lattice system, the observed peak spacing as $\vec{q}^2_{obs}$, is scaled using a standard scaler. Meaning the mean and standard deviation of $\vec{q}^2_{obs}$ for all entries in the training set are calculated. Then the mean is subtracted from each peak spacing and the difference is divided by the standard deviation. All of the peaks are scaled with the same mean and standard deviation regardless of the order in the peak list.

Unit cell length parameters are also scaled using a standard scaler. For a given lattice system, the mean and standard deviation of all the unit cell lengths of the training set entries are calculated and used for scaling.

Unit cell angle parameters are converted to radians and subtracted by $\pi/2$. This puts right angles (90^o) at zero. Then the standard deviation of all angles, which were not originally 90^o, is calculated in radians, and the zero-centered unit cell angles are divided by this scale. *On my refactoring todo list is convert the angle scaling to just cosine of the angle.*

2.1.5 Reference Miller Indices

This work treats Miller indices as sets of three integers. These are categorically discrete variables that are assigned nominal labels based on the uniqueness of the peak position they would produce for any unit cell. Reference sets Miller indices are created for each Bravais lattice. First a large reference set with h, k, and l taking on all possible values from -40 to 40 is created. This list is first reduced to eliminate redundancy from lattice system constraints, to create a set of Miller indices that produce unique peak spacings for any unit cell of the given lattice system. These are given by the following formulas:

- Cubic lattice system

  For each Miller index in the large reference array, $h^2 + k^2 + l^2$ is calculated. For each unique value of $h^2 + k^2 + l^2$, one set of Miller indices is retained. These are then sorted in order of increasing $h^2 + k^2 + l^2$.

- Tetragonal lattice system

  $h^2 + k^2$, and l^2 are calculated. For each unique set of $h^2 + k^2$ and l^2, one set of Miller indices is retained. These are then sorted in order of increasing $h^2 + k^2 + l^2$.

- Hexagonal lattice system $h^2 + hk + k^2$, and l^2 are calculated. For each unique set of $h^2 + hk + k^2$, and l^2, one set of Miller indices is retained. These are then sorted in order of increasing $\frac{4}{3}(h^2 + hk + k^2) + l^2$.

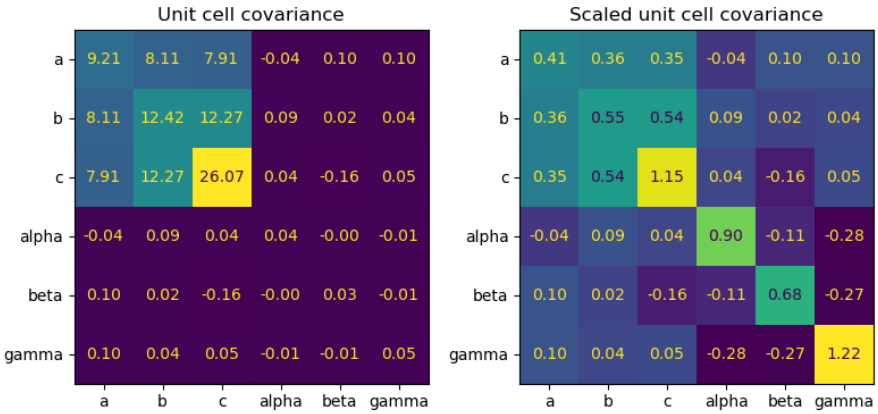


Figure 3: Covariance of the unit cell parameters for the triclinic lattice system. The left column is the covariance matrix for the raw unit cell parameters. The right column is the covariance after standard scaling has been applied.

- Rhombohedral lattice system $h^2+k^2+l^2$ and $hk+kl+hl$ are calculated. For each unique set of $h^2+k^2+l^2$ and $hk+kl+hl$, one set of Miller indices is retained. These are then sorted in order of increasing $h^2+k^2+l^2$.
- Orthorhombic lattice system For each Miller index in the "all possible set", h^2, k^2, and l^2 are calculated. Each unique set of h^2, k^2, and l^2 results in a unique peak. These are then sorted in order of increasing $h^2 + k^2 + l^2$.
- Monoclinic lattice system h^2, k^2, l^2, and hl are calculated. For each unique set of h^2, k^2, l^2, and hl, one set of Miller indices is retained. These are then sorted in order of increasing $h^2 + k^2 + l^2$.
- Triclinic lattice system h^2, k^2, l^2, hk, kl and hl are calculated. For each unique set of h^2, k^2, l^2, hk, kl and hl, one set of Miller indices is retained. These are then sorted in order of increasing $h^2 + k^2 + l^2$.

These reference sets are further reduced by eliminate Miller indices that are systematically absent for the given Bravais lattice.

The last step to establish the reference set of Miller indices is a final sorting. The peak spacing is calculated for each of peak in the set of unaugmented training data given the reference Miller index. The average peak spacings are then averaged over all the entries. Then the reference Miller index set is sorted so these averaged peak spacings increase monotonically. Each reference set is then reduced to fixed length that differs from each set. Generally it is set so at least 99% of entries have all their observed Miller indices in their corresponding reference set. The Miller indices taken from the peak list are then assigned a nominal label corresponding to the position of the equivalent Miller index in the sorted reference list. If a peak's Miller index is not found in the reference list, which occurs because the equivalent Miller index is beyond the 100 or 500 length cutoff, it is assigned the last nominal label which is given a Miller index of (000).

2.1.6 Augmentation

The grouping approach requires a broad diversity of data, as opposed to just simply more data. A broader range of data allows for more groups. To increase the breadth of data, augmentation is performed, preferentially augmenting entries associated with Hermann–Mauguin spacegroup symbols with fewer entries. Augmentation happens after splitting the dataset into training and validation sets. If the original entry is in the training set, all entries augmented from this entry will remain in the training set and visa-versa for the validation set. A single entry is augmented at most 10 times.

Two augmentations need to be performed for each entry. First, the unit cell parameters are randomly perturbed. Second, the observed peaks are resampled from the non-systemically absent peaks. There are three implemented methods to randomly perturb the unit cell parameters, all of these perturbations occur in the scaled unit cell parameter space:

1. Random perturbation. A unit cell parameter for an entry is perturbed by randomly sampled from a Normal distribution with the original entry's unit cell parameter as a mean and 0.2 as the standard deviation. This perturbation occurs in the scaled unit cell space, so it is much larger than 0.2Å.

2. Covariance perturbation. The covariance matrix of the scaled unit cell parameters is calculated over all the training entries within a lattice system. Scaled unit cell parameters for an entry are perturbed by sampling from a multivariate normal distribution centered at the unperturbed scaled unit cell parameters and with a covariance matrix equal to 0.2^2 times the covariance matrix.

3. SVD perturbations. An SVD is performed on the scaled unit cell parameters from the training set within a lattice system. The training set's unit cell parameters are then transformed into the new orthogonal basis from the SVD and the standard deviation of each component is calculated. An entry's unit cells are perturbed by transforming the scaled unit cell parameters using the SVD, then sampling new components from a Normal distribution centered at the entry's transformed component values and with a standard deviation equal to 0.2 times the standard deviation of that component calculated over the training set.

The random perturbation method is used only for the cubic lattice system because a covariance or SVD is not possible with one parameter. All lattice systems other than cubic use the SVD perturbation method. The covariance and SVD methods were implemented so any correlation between unit cell parameters observed in the training data would be retained in the augmented entries. The covariance method is implemented in the code, but not used.

The peak list is augmented by resampling peaks from the list of non-systematically absent peaks. This resampling method was developed so the distribution of peaks in the augmented entries matched the distribution of the unaugmented entries.

There are two peak lists available at this time; Peak list 1, the first 60 non-systematically absent peaks. Peak list 2, the first 60 peaks that could be realistically observed in a diffraction pattern. Again, only using the training data to set up this augmentation. The position of the first observed peak in the list of all non-systematically absent peaks is recorded. An empirical distribution is made from a histogram of this position list. For the augmented entries, the empirical distribution is used to randomly select a first peak from the list of all non-systematically absent peaks.

After the first peak is picked, the list of non-systematically absent peaks is stepped through and the next peak is picked based on its distance from its neighboring peaks. Selecting the next peak based only on its separation is chosen to make observability only based on whether or not the peak will overlap another peak. The intensity of the peak is not a consideration. This approach is set up by calculating the minimum separation, Δq^2 in units of q^2, between each peak in Peak list 1 and its neighbors. The fraction of neighboring peaks in Peak List 2 that are also neighboring in Peak List 1 are calculated in binned units of Δq^2. This binning is in 50 bins log spaced between 10^{-3} to 10^{-2} for cubic and 10^{-4} to 10^{-2} otherwise. A function of the form,

$$\rho(\Delta q^2, q^2) = c(q^2)\left[1 - r_0^{-r_1 \Delta q^2}\right]. \tag{19}$$

is fit to the binned fraction and is used as and acceptance probability. This form was chosen because when $\Delta q^2 = 0$ the probability of acceptance is zero and as $\Delta q^2 =>$ inf, the probability of acceptance approaches one. $c(q^2)$ is a linear function of q^2. *c(q2) should be bounded above by 1.* It should be noted that $\rho(\Delta q^2, q^2)$ remains less than one in the observed range of Δq^2 so a well separated peak can be rejected. The probability increases monotonically between these two extremes.

After the first peak is selected, there are three different cases for peaks:

1. Peak is within 0.1^o / 1.5 of the preceding selected peak. Reject peak.

2. Peak is further than 0.1^o / 1.5 of the preceding selected peak and the next peak in Peak list 1. Accept peak with probability $\rho(\Delta q^2)$.

3. Peak is further than 0.1^o / 1.5 of the preceding selected peak and with 0.1^o / 1.5 of the next peak in Peak list 1. Accept peak with 50% probability. If rejected, select the next peak. This assumes that if two peaks overlap, only one will be observed. Which peak is completely random.

The peak resampling method is a bit complicated. I arrived at it by trying to ensure the distribution of Miller indices in the augmented set matched the distribution in the unaugmented set.

2.2 Unit cell generator models

Unit cell generator models are developed to use their outputs to randomly generate unit cell candidates based on a set of peaks. For all lattice systems except cubic, 20 peaks from peak list 2 are used. For cubic lattice systems, 10 peaks are used because there tend to be fewer observable peaks in the diffraction patterns and 10 peaks are sufficient for one free parameter.

These models should be evaluated based on two criteria, robustness and efficiency. Given a neighborhood near the true unit cell parameters, robustness is the abiliity of a model to generate a minimum number of

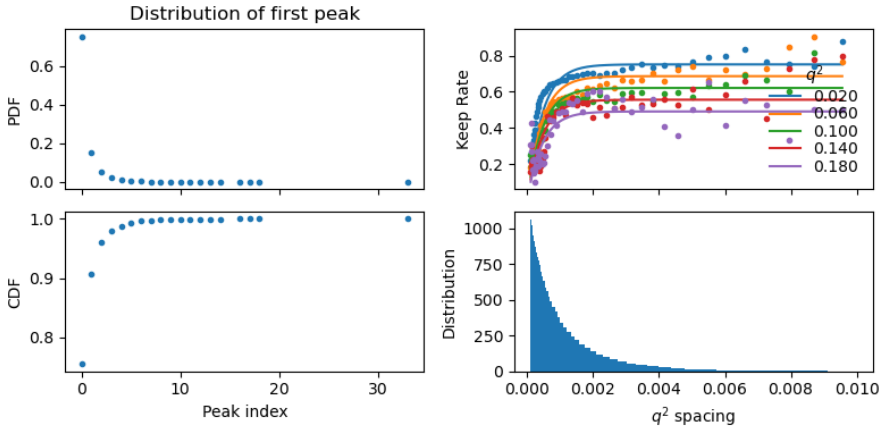


Figure 4: Augmentation statistics for the triclinic lattice system determined from the training entries. The left column is the probability distribution for the index of the first observed peak in the list of all non-systematically absent peaks. The top left plot is the PDF and bottom plot is the CDF. The lower right column plots the distribution of spacings between all possible non-systematically absent peaks. The upper right plot is the probability that a peak is observed vs. the separation from its closest neighboring peak. The different colors represent different ranges of absolution peak position. The markers are the observed data and the solid lines are the fit curves.

candidate unit cells within the neighborhood and efficiency is the fraction of generated entries within the neighborhood. Robustness is the most important quality as candidates must be generated near the correct value in order for a solution to be found.

2.2.1 Neural Networks

Don't read too much into this section. I wouldn't be surprised if we cut the neural network models out entirely as they seem to be the worst performers.

Regression neural networks are set up as variance networks (Nix & Weigend, 1994). They output a mean and variance and are trained using a negative log-likelihood target function, which assumes a Normal likelihood. The approach presented here is based heavily on recent research on Variance networks (Detlefsen, et. al., 2019; Seitzer, et. al., 2022; Stirn & Knowles, 2021; Stirn et. al., 2022). Features implemented from this work include:

1. A distribution is predicted for variance as opposed to a point estimate. The Inverse-Gamma distribution is the posterior distribution for the unknown variance of a Normal distribution and is parameterized by α_{σ^2} and β_{σ^2}. The neural network outputs three values for each unit cell parameter, mean= μ, α_{σ^2} and β_{σ^2}.

2. Each model contains three parallel neural networks outputing μ, α_{σ^2} and β_{σ^2} seperately. These networks start independently from the input peak spacings are have no connections with each other.

3. The training occurs cyclically. First, the weights of the μ network are optimized while the weights of the α_{σ^2} and β_{σ^2} networks are held fixed. Then the weights of the μ network are fixed while the weights of the α_{σ^2} and β_{σ^2} networks are optimized. Generally, each cycle is run for 10 epochs, and five cycles are performed.

4. A β-NLL target function is used when training the μ network and an NLL target function is used when training the α_{σ^2} and β_{σ^2} networks. The NLL target function is defined as:

$$L_{NLL} = \Sigma_{i=0}^{n_{uc}} \frac{1}{2}\ln(2\pi) - \alpha_{\sigma^2,i}\ln(\beta_{\sigma^2,i}) - \ln(\Gamma(\alpha_{\sigma^2,i}+1/2)) + \ln(\Gamma(\alpha_{\sigma^2,i})) + (\alpha_{\sigma^2,i}+1/2)\log(\beta_{\sigma^2,i}+\frac{1}{2}(uc_i - \hat{uc}_i)^2). \tag{20}$$

Here, the summation occurs over the unit cell parameters, uc_i are the true unit cell parameters, and $\hat{uc}_i$ is the estimated mean. The β-NLL target function multiplies the NLL target function by the variance estimate raised to the power β_L. This multiplicative term is excluded from the gradients calculations, as denoted by the stop-gradient operator $\lfloor\cdot\rfloor$,

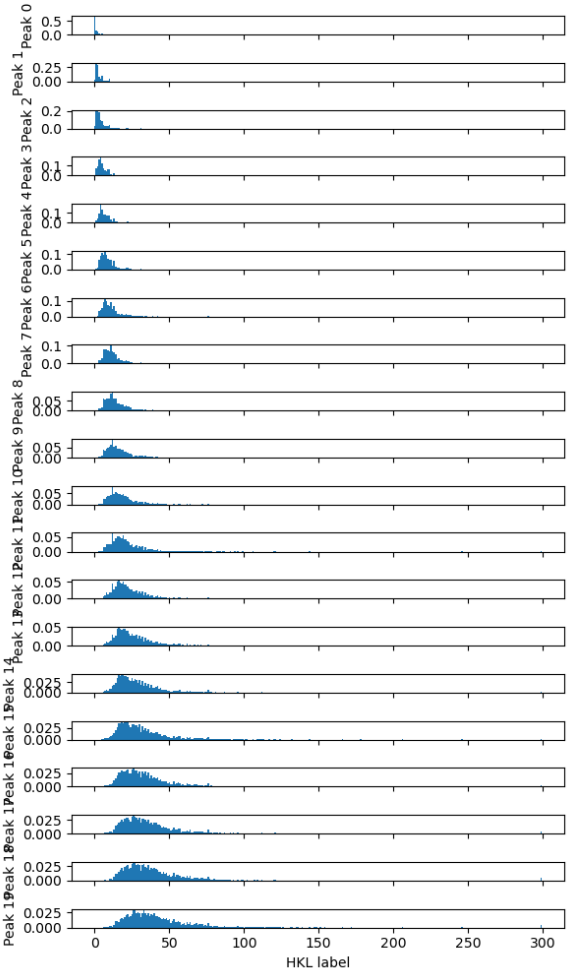


Figure 5: Unaugmented

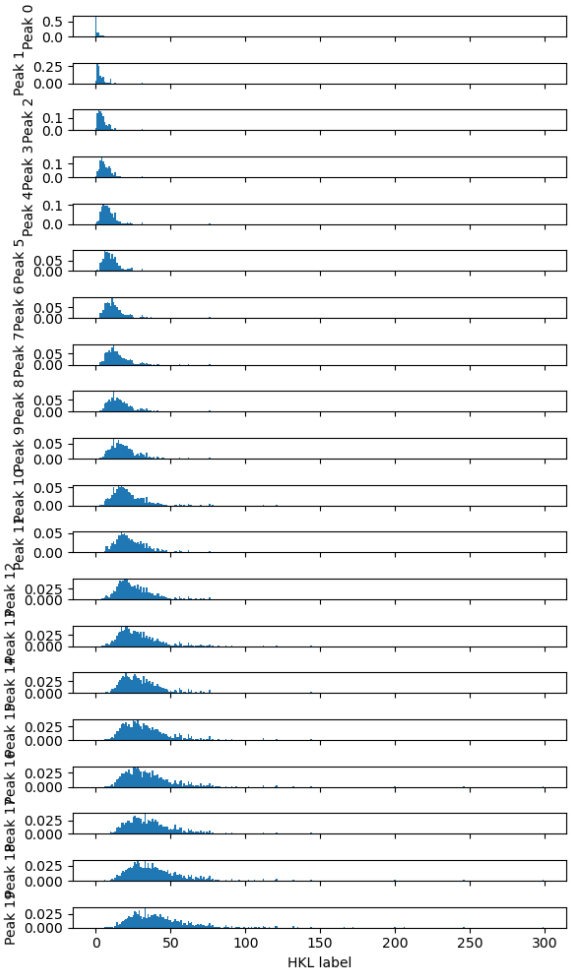


Figure 6: Augmented

Figure 7: Distribution of nominal Miller set labels for triclinic. These can be used for diagnostics of the augmentation process. The distributions should look the same for the augmented and unaugmented entries.

$$L_{\beta-NLL} = \Sigma_{i=0}^{n_{uc}} \left\lfloor \left(\frac{\beta_{\sigma^2,i}}{\alpha_{\sigma^2,i} - 1} \right)^{\beta_L} \right\rfloor \left[\frac{1}{2} \ln(2\pi) - \alpha_{\sigma^2,i} \ln(\beta_{\sigma^2,i}) - \ln(\Gamma(\alpha_{\sigma^2,i} + 1/2)) + \ln(\Gamma(\alpha_{\sigma^2,i})) + (\alpha_{\sigma^2,i} + 1/2) \log(\beta + \frac{1}{2}(u \right. \tag{21}$$

In the Inverse-Gamma distribution estimate of variance, the variance is estimated as $\frac{\beta_{\sigma^2}}{\alpha_{\sigma^2}-1}$ where $\alpha_{\sigma^2} > 1$. The β-NLL target function raises this value to the power of $\beta_L = 1/2$ and multiplies the standard NLL target function.

I do not know which of the four of these points are important or not. There are still a few recommendations in the four papers referenced to improve the variance estimates that could be implemented.

There is an insidious memory leak in Tensorflow. I need to recompile the models between each cycle as the fixed weights and target function changes. Every time the model recompiles, a new "graph" is created in the backend of Tensorflow and the old graph is not released from memory. This is extremely bad when I am training 10 - 20 models per lattice system and I need to recompile the model 10 times during the training. I have spent some time trying to fix it without resolution. This is not the first time I have struggled with Tensorflow memory leaks and feel like jumping ship to Pytorch might be justified to solve this issue.

Each of the networks separately predicting μ, α_{σ^2} and β_{σ^2} are dense networks (MLP, Feed-forward). The default mean network uses three hidden layers of 200, 100, and 60 units. These hidden layers proceed in the following order, 1) Dense layer, 2) Layer Normalization, 3) GELU activation, 4) Dropout at a rate of 50%. The final layer is a Dense layer that outputs with linear activation.

The default α_{σ^2} and β_{σ^2} networks use two hidden layers of 100 and 60 units and are otherwise similar to the mean network. The final layer of the α_{σ^2} and β_{σ^2} networks uses a softplus activation to enforce a positive constraint on β and one is added to the α_{σ^2} network after softplus activation to enforce an $\alpha_{\sigma^2} > 1$ constraint.

All training occurs with the Adam optimizer using a learning rate of 0.0001 and a batch size of 64. The Neural networks are implemented in TensorFlow.

I have not done any hyperparameter optimization. There is a lot of over and underfitting.

2.2.2 Random Forest

In addition to the neural networks described in Section 2.2.2, random forest regression models are fit for each group of data. These are implemented using SKlearn with the parameters: 80 decision trees each trained with a 10% subsample of the training set and a minimum of 10 samples per leaf. Otherwise, the parameters were the defaults of *sklearn.ensemble.RandomForestRegressor*.

Some hyperparameter optimization was performed on the triclinic lattice system. The triclinic random forest model used 2,000 decision trees each trained with a 5% subsample and a minimum of 1 sample per leaf.

2.2.3 Miller Index Templates

The last model for unit cell generation is based on randomly generating template sets of Miller indices that can be used to analytically determine unit cells given a set of observed peak spacings. The templates are genearted based on the observed distribution of Miller indices in the training set. To ensure robustness, there must be a balancing to prevent over-representation of more common unit cells. This balancing is performed to balance "dominant zones".

A crystal with a dominant zone is a typically defined as having one axis that is relatively shorter than the other two. Diffraction peaks that contain information about this shorter axis will occur at relatively larger diffraction angles. This has two distinct effects on the powder diffraction pattern that affect the presence and content information about the short axis. First, there relatively few peaks in the diffraction pattern that contain information about the short axis, and specifically, these peaks will not occur at low diffraction angles. Secondly, for peaks that do contain information about the short axis in addition to the other axes, the Miller index corresponding to the short axis will be numerically smaller.

We use two metrics to quantify dominant zones that target these two effects separately. The first metric targets the lack of information about one axis in the diffraction pattern, the second targets the distribution of Miller indices for the short axis. This distinction is important, because the lack of information about one axis could be caused by unbalanced systematic absences along each axis or by unlucky chance of randomly unobserved peaks, in addition to differences in diffraction angles caused by differences in lattice length. For example, we reindex all base-centered monoclinic entries to the body-centered setting. In the transformation that converts a conventional c-centered monoclinic to the body-centered setting, the lattice vectors are transformed via $\vec{a}' = \vec{c}$, $\vec{b}' = \vec{b}$, $\vec{c}' = -\vec{a} - \vec{c}$. Generally speaking, this should increase the magnitude of $\vec{c}$ to $\vec{a}$. At first glance, this will increase the relative information about the c-axis making the "dominant zone" problem worse. However, the reflection conditions change from $h + k = 2n$ to $h + k + l = 2n$, introducing the $l = 2n$ condition when

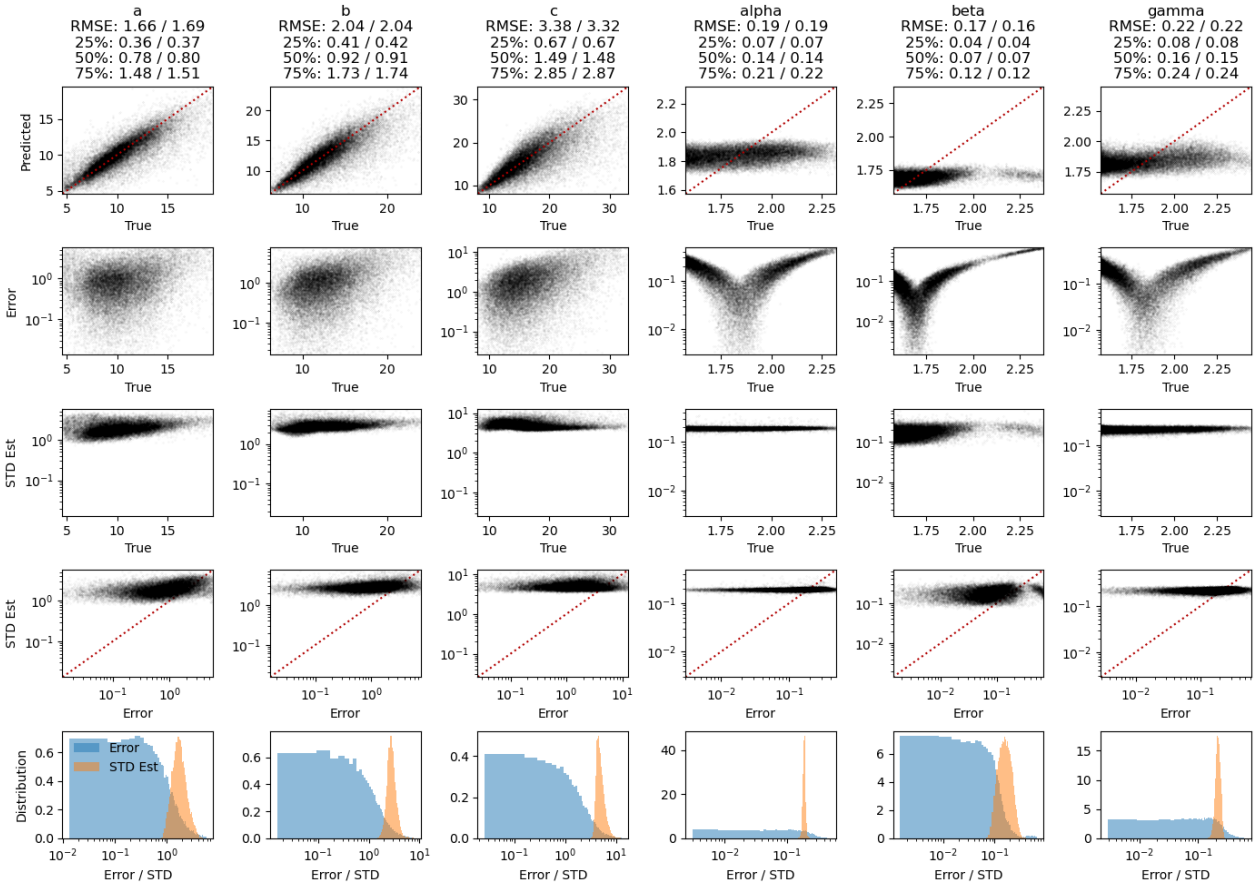


Figure 8: Regression results for the neural network model. The top row plots the predicted vs true unit cell parameters. The second row plots prediction error vs true parameters. The titles print the RMSD and percentiles for the training and validation sets.

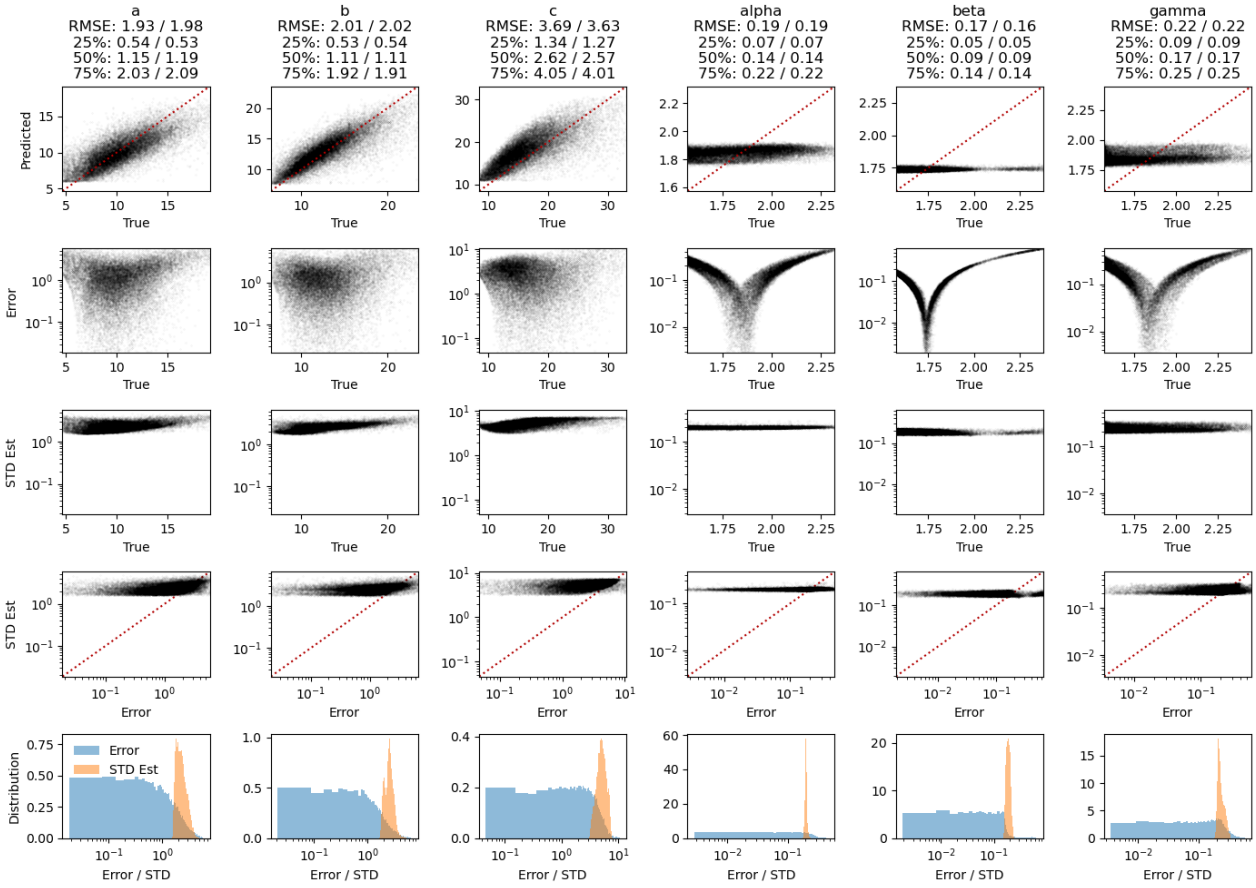


Figure 9: Regression results for the random forest model. The top row plots the predicted vs true unit cell parameters. The second row plots prediction error vs true parameters. The titles print the RMSD and percentiles for the training and validation sets.

$(hkl) = (00l)$, which cancels out the first dominant zone effect of presence of information, but not necessarily the content of information effect.

The first metric is a new metric that quantifies the minimum amount of information about a single axis. For this metric, the number of peaks that contain information about each axis, the number of peaks with a non-zero Miller index, are calculated. Then the minimum of the three axes is taken. For each the three peak list $(001), (011), (111)$ contains one, two and three bits of information of information about axes a, b, and c respectively. This diffraction pattern would then be a one bit pattern.

The second metric is the ratio of the shortest to longest unit cell axis.

To start the Miller index template generation, the entries are separated into bins based on the information content (21 bins with values of 0 - 20 when there are 20 peaks). Within each bin, N templates are generated. If there are less than N entries in the bin, each entry is directly taken as a templating. Otherwise, templates are randomly generated based on the distribution of Miller indices in the training set. In this case, a histogram of the first peak's Miller set is taken as a probability distribution. The template's first Miller set is randomly sampled from this distribution. For the second peak, a histogram is created for the second peak's Miller set, given its first peak has the same Miller set as the random selection. The second Miller index is sampled from this distribution. This algorithm is repeated until each of the template's peaks are assigned or a histogram is attempted to be created from a single entry. In the latter case, the remaining entries Miller sets are used for the template's remaining peaks.

This is procedure is then repeated based on bining on the second metric. Ten bins are created that span 0-1.

In the case that there are fewer than N entries in the bin, the ratio of N to the number of entries is recorded. Later, if a random subsampling of the peaks is performed, the ratios are normalized and used as sampling probabilities.

To generate a unit cell from the templates, first a random template is selected, then an optimization algorithm is performed to determine the unit cell:

1. Set $\vec{w} = 1/\vec{q}^2_{obs}$.
2. Solve a weighted version of Eq. 17, $\vec{w}\vec{q}^2_{obs} = \vec{w}H\vec{x}_{nn}$. This is performed unit *numpy.linalg.lstsq* which solves the equation using the SVD method.
3. Calculate the forward model, $\vec{q}^2_{calc} = H\vec{x}_{nn}$. If $\vec{q}^2_{calc}$ monotonically increases, accept $\vec{x}_{nn}$, otherwise,
4. Reorder the Miller index template to produce a sorted $\vec{q}^2_{calc}$.
5. Set $\vec{w} = 1/\sqrt{\vec{q}^2_{obs}|\vec{q}^2_{obs} - \vec{q}^2_{calc}|}$.
6. Repeat starting at step 2 until a sorted $\vec{q}^2_{calc}$ is obtained or 10 iterations have been performed.
7. Record a loss $L = |1 - \vec{q}^2_{calc}/\vec{q}^2_{obs}|$.

After all the Miller index templates are generated, only unique templates are retained.

The loss is not currently used. Eventually I would like to use this as a metric to improve this process. The idea would be to view this templating method as a "feature extraction" layer. Combine metrics from the unit cell optimization and use as inputs to a random forest / neural network model that gives a sampling probability for the templates.

2.3 Miller Index Assignment Model

The second primary component of an indexing algorithm is the assignment of Miller indices given a unit cell and observed peak spacings. This is performed by establishing a reference set of Miller indices. These are all the possible Miller indices for a given lattice system without consideration for systematic absences. The forward diffraction models, (Eq. 1-7) are used to calculate a reference set of peak spacings from this reference set of Miller indices and a given unit cell. A pairwise difference array is then calculated between the observed and reference set of peak spacings. This array is then used as inputs to a neural network which produces a probability distribution, for each observed peak, for the correct Miller index assignment over the reference set of Miller indices.

2.3.1 Neural Network

The inputs of the assignment neural network are scaled unit cell parameters and scaled $\vec{q}^2_{obs}$. Pairwise differences are calculated by first calculating the scaled peak spacing for each reference Miller index given the input unit cell parameter, $\vec{q}^2_{ref}$. This is subtracted from the observed peak spacing in a pairwise manner, $\Delta_{\vec{q}^2} = \vec{q}^2_{obs} - \vec{q}^{2\,T}_{ref}$. If there are 20 input peaks and a reference peak list of 500, this will result in a 20 x 500 array. If the

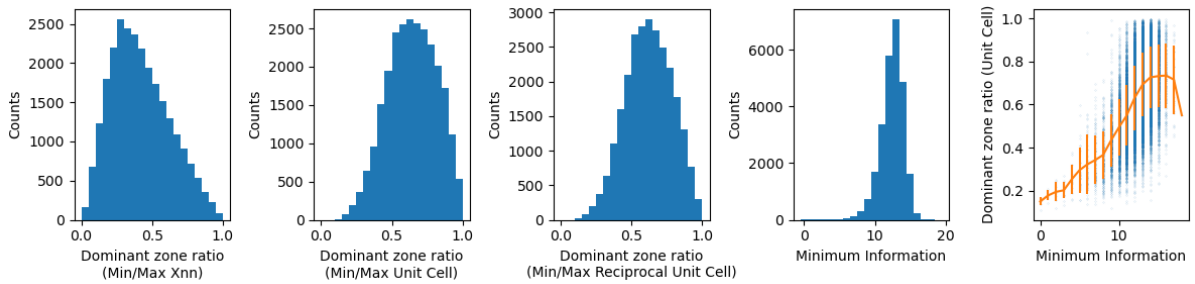


Figure 10: Dominant zone metrics for triclinic. The left three histograms are metrics for dominant zones based on unit cell values. These are quantified for the ratios of the x_{nn} parameters, the direct unit cell, and the reciprocal unit cell. The fourth plot is the minimum information. The rightmost plot is a scatter plot of dominant zone ratio vs minimum information. While these are correlated on average (orange line), there is considerable deviation from average.

pairwise differences are used to predict peak positions, the correct assignment will occur at small values of $\Delta_{\vec{q}^2}$. Additionally, incorrect assignments will occur at large absolute values of $\Delta_{\vec{q}^2}$. $\Delta_{\vec{q}^2}$ is transformed so it will span the range of 0 to 1, where the correct assignment will occur at larger values,

$$\Delta^*_{\vec{q}^2} = \frac{\epsilon_\Delta}{|\Delta_{\vec{q}^2}| + \epsilon_\Delta}. \tag{22}$$

$\Delta^*_{\vec{q}^2}$ is used as the input to a dense neural network. The default network uses four hidden layers of 200, 200, 200, and 100 units. These hidden layers proceed in the following order, 1) Dense layer, 2) Layer Normalization, 3) GELU activation, and 4) Dropout at a rate of 25%. These hidden layers operate on two-dimensional arrays of shape (# peaks, # reference Miller indices) for the first hidden layer and (# peaks, # previous layer units) for the second hidden layer. The Dense layers in Tensorflow operate on these 2D inputs as follows.

1. Create a kernel with shape (# previous layer units, # current layer units).
2. For each peak dimension, operate on the input array with the same kernel from step 1).

In other words, the same kernel operates on each peak dimension in the hidden layer independently. I believe the implications of this are 1) information isn't being shared between peak dimensions, and 2) Each peak dimension is considered equivalently.

The final layer is a dense layer that outputs with softmax activation. Instead of one dense layer output, the network is branched for each peak and a separate dense layer - meaning a separate kernel - acts on the one-dimensional input for the given peak.

An important aspect of this NN is fast inference. I would like to find the smallest possible NN that provides reasonable results. Some testing shows that I can move to a one hidden layer NN with minimal loss of accuracy It is tempting to try to use a convolutional neural network here. I did see improved accuracy with cubic. In that case, the reference set of Miller indices increases with q2. There is a well defined spatial relationship in the pairwise difference array. This is lost in the other lattice systems.

An important question is, what data should these networks be trained with? One option is the true unit cell parameters, this results in 100% accurate Miller index assignments (and it does). On the other hand, they could be trained using the predicted unit cell parameters from the regression networks. This should represent a worst-case scenario for unit cells. There should be a few randomly generated unit cells better than the expected value. After several rounds of optimization, the best candidate unit cells should be much better than their initial value. Ideally, the Miller index assignment networks would be trained on unit cell parameters that differ from the true values by amounts consistent with the current best candidate unit cells during optimization. To these means, a series of assignment networks are trained with input unit cell parameters randomly perturbed from their true values. This is performed by adding Gaussian noise to the scaled unit cell parameters before they are used to calculate pairwise differences. This is set up in a way that each entry receives a different perturbation for each training epoch for additional regularization.

Training of the assignment neural networks is performed with all the unaugmented entries for a given lattice system. Augmented entries are omitted because there is plenty of unaugmented data to train the networks - and I have a bug somewhere in the augmentation algorithm regarding the Miller index assignments. The sparse categorical cross-entropy target function is chosen and the Adam optimizer is used with a learning rate of 0.002 and a batch size of 256. Networks are trained with Gaussian noise levels of 0.10, 0.05, 0.025, and 0.01. For monoclinic, the associated ϵ_Δ values were 0.01, 0.0075, 0.01, and 0.005.

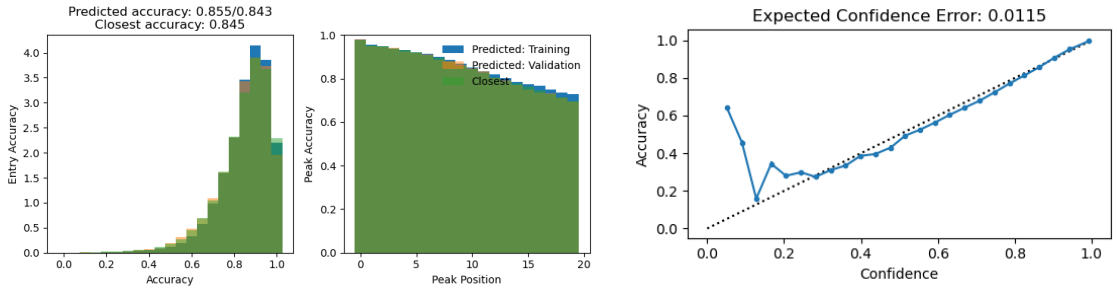


Figure 11: Assignment diagnostics for triclinic with 0.5% unit cell perturbation. The left plot is the accuracy and the right plot is the calibration. The accuracy plots a histogram of accuarcy for entire entries. The blue plot is the training data and the orange plot is the validation data. The green plot is a baseline that represents the accuracy if the closest peak was chosen instead of the neural network predictions. The calibration plot shows the average accuracy vs confidence.

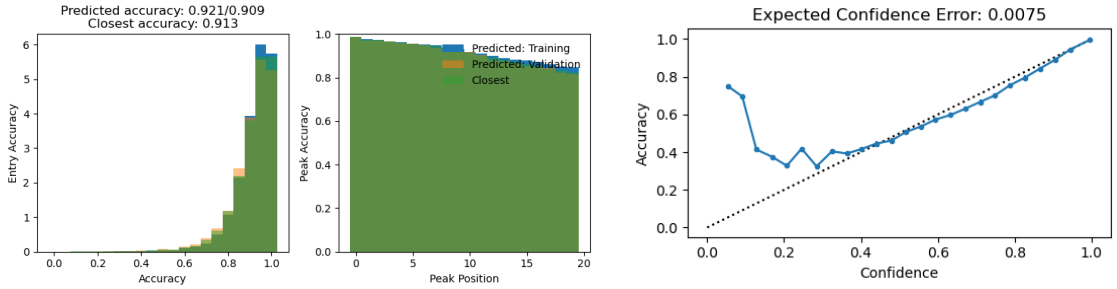


Figure 12: Assignment diagnostics for triclinic with 0.25% unit cell perturbation.

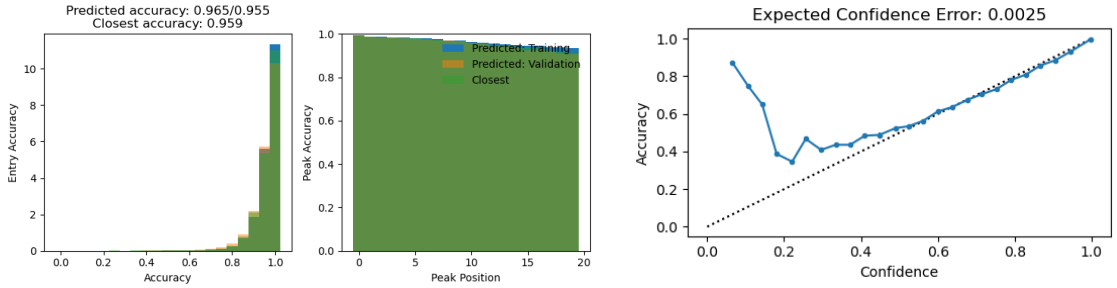


Figure 13: Assignment diagnostics for triclinic with 0.1% unit cell perturbation.

2.4 Optimization

The third component of the indexing algorithm is the optimization of unit cell parameters given initial unit cell candidates and a mechanism to assign Miller indices. Currently, the program only considers one lattice system at a time. So if a crystal is known to be monoclinic, it will be optimized with that as a known fact. After getting this working with all lattice systems, I will work on the case where the lattice system is unknown.

2.4.1 Target function

The optimization seeks the most probable Miller index, H, and unit cell, $\vec{c}$, given a set of observed peak spacings, $\vec{q}_{obs}$. This can be expanded using Baye's formula.

$$\rho(H, \vec{c}|\vec{q}_{obs}) = \frac{\rho(\vec{q}_{obs}|H, \vec{c})\rho(H, \vec{c})}{\rho(\vec{q}_{obs})}. \tag{23}$$

The denominator can be dropped because it will be constant for all unit cell parameters and Miller index assignments,

$$\rho(H, \vec{c}|\vec{q}_{obs}) \propto \rho(\vec{q}_{obs}|H, \vec{c})\rho(H, \vec{c}). \tag{24}$$

Miller index assignments are conditional on the unit cell parameters,

$$\rho(H, \vec{c}|\vec{q}_{obs}) \propto \rho(\vec{q}_{obs}|H, \vec{c})\rho(H|\vec{c})\rho(\vec{c}). \tag{25}$$

This can be written as a product of probabilities over each observed peak spacing $q_{obs,k}$, assuming there are n_k peaks and independence between peaks.

$$\rho(H, \vec{c}|\vec{q}_{obs}) \propto \Pi_{k=1}^{n_k} \left[\rho(q_{obs,k}|H_k, \vec{c})\rho(H_k|\vec{c})\right] \rho(\vec{c}). \tag{26}$$

This equation can be further expanded as a summation over all n_h reference Miller indices for each of the n_k peaks,

$$\rho(H, \vec{c}|\vec{q}_{obs}) \propto \Pi_{k=1}^{n_k} \left[\Sigma_{h=0}^{n_h} \rho(q_{obs,k}|H_{k,h}, \vec{c})\rho(H_{k,h}|\vec{c})\right] \rho(\vec{c}). \tag{27}$$

From here, this likelihood function is simplified. The probability distribution of the unit cell lengths is assumed to be constant and bounded between 2 and 500 $\mathring{A}$. The probability distribution of the unit cell angles is assumed to be constant and bounded between 0^o and 180^o. The term $\rho(\vec{c})$ is removed from the likelihood function and out-of-bounds unit cells are removed from consideration,

$$\rho(H, \vec{c}|\vec{q}_{obs}) \propto \Pi_{k=1}^{n_k} \left[\Sigma_{h=0}^{n_h} \rho(q_{obs,k}|H_{k,h}, \vec{c})\rho(H_{k,h}|\vec{c})\right]. \tag{28}$$

On my to-do list is to optimize this term directly. It is entirely possible with the current set of infrastructure. The Miller index neural networks are differentiable. I can optimize the unit cell parameters considering all possible Miller Index assignments. Or put it into a Tensorflow Probability MCMC method. Until then, I am only considering one at a time for simplification.

The likelihood equation is again simplified by only considering the j^{th} Miller index assignment at a time. The most obvious choice is the most probable Miller index assignment, but this is not necessarily the case.

$$\rho(H, \vec{c}|\vec{q}_{obs}) \propto \Pi_{k=1}^{n_k} \rho(q_{obs,k}|H_{k,j}, \vec{c})\rho(H_{k,j}|\vec{c}). \tag{29}$$

The term $\rho(H_{k,j}|\vec{c})$ is the softmax output from the assignment neural network. The term $\rho(q_{obs,k}|H_{k,j}, \vec{c})$ is a normal distribution,

$$\rho(q_{obs,k}|H_{k,j}, \vec{c}) = \frac{1}{\sqrt{2\pi}\sigma_{q^2}} \exp\left(-\frac{1}{2}\frac{(q^2_{obs,k} - q^2_{cal,kj})^2}{\sigma^2_{q^2}}\right). \tag{30}$$

Eq. 1-7 are used to calculate $q^2_{cal,kj}$. The standard deviation parameter, $\sigma^2_{q^2}$, is defined like the weighting by SVD-Index,

$$\sigma^2_{q^2} = q^2_{obs,k}|q^2_{obs,k} - q^2_{cal,kj} + \epsilon_{q^2}|. \tag{31}$$

Notably, the $\sigma^2_{q^2}$ presented here does not include the randomness in the same way as the weighting in SVD-Index. Randomness is important to the algorithm and is included by other means.

2.4.2 Random Unit Cell Generation

Random unit cells are generated from the neural networks. For each entry, a set of mean and variance estimates are taken from each neural network trained on the different groups. Each set of mean and variance estimates are used as parameters for a Normal distribution and n_{nn} random unit cell parameters are generated. n_{rf} random unit cells are also selected from individual decision trees in the random forest model. The template model is used to generate n_{temp} unit cells.

Once random unit cells are generated, there will be $n_g(n_{nn} + n_{rf}) + n_{temp}$ candidate unit cells where n_g is the number of groups.

Model	Entries unable to place one close candidate	Efficiency
Neural Network	306	2.28%
Random Forest	206	2.52%
Templates	87	2.57%
Neural Network & Random Forest	137	2.39%
Neural Network & Templates	72	2.42%
Random Forest & Templates	54	2.54%
All	49	2.45%

Table 3: Evaluation of robustness and efficiency of random unit cell generators. For each unit cell generation model, 2,000 candidates were generated and the number of entries where the models failed to place one candidate within a 1 Å of the true unit cell is recorded as the middle column. The percentage of candidates within this region is in the right column. This evaluation was performed using roughly 5,000 triclinic entrices

2.4.3 Randomized Optimization

The candidates are optimized using a random search procedure. The general approach is to iteratively assign Miller sets and update unit cells with randomization applied to the Miller set assignment. There are different randomization methods and the ideal method has not been determined.

No randomization The first approach to randomization is to not include it. This is included to explain the general approach.

1. Given the current unit cell estimate, calculate the pairwise difference array, $\Delta_{\vec{q}^2}$. For each peak position, assign the closest peak, $argmin(|\Delta_{\vec{q}^2}|)$. For all of the Miller set assignment algorithms discussed here, assume that there is a mechanism that prevents the same Miller set to be assigned to multiple peaks.

2. Update the unit cells given the new Miller set assignments. This could be performed by solving the linear model of Eq. 17. Instead this is treated as a nonlinear least squares and the unit cells are updated with a single Gauss-Newton step calculated from the negative log likelihood, where the likelihood function is specified in Eqs. 30 & 31.
$$\vec{x_{nnt+1}} = \vec{x_{nnt}} + \mathcal{H}_t^{-1} \nabla \ln[\rho(\vec{q}_{obs}|H_t, \vec{x_{nnt}})] \tag{32}$$
Here, $\mathcal{H}_t^{-1}$ is the inverse Hessian of the negative log likelihood and ∇ is the gradient operator.

3. Recalculate the negative log-likelihood given the current unit cell and Miller set assignments.

4. Ensure that the unit cell is within a physically realistic range. If not fix the unit cell.

5. If the negative log-likelihood is below a specified threshold, it is determined to explain the observed diffraction and retained. The threshold value is based on the numerical value when $q^2_{cal} \approx q^2_{obs}$

$-\ln\left[\Pi_{k=1}^{n_k}\rho(q_{obs,k}|H_j,\vec{c})\right] = -\Sigma_{k=1}^{n_k}\ln\left[\frac{1}{\sqrt{2\pi}\sigma_{q^2}}\exp\left(-\frac{1}{2}\frac{(q^2_{obs,k}-q^2_{cal,k})^2}{\sigma^2_{q^2}}\right)\right]$,

$= -\Sigma_{k=1}^{n_k}\ln\left[\frac{1}{\sqrt{2\pi}\sigma_{q^2}}\right] + \ln\left[\exp\left(-\frac{1}{2}\frac{(q^2_{obs,k}-q^2_{cal,k})^2}{\sigma^2_{q^2}}\right)\right]$,

$= \Sigma_{k=1}^{n_k}\ln\left[\sqrt{2\pi}\sigma_{q^2}\right] - \frac{1}{2}\frac{(q^2_{obs,k}-q^2_{cal,k})^2}{\sigma^2_{q^2}}$,

Now given $\sigma^2_{q^2} = q^2_{obs,k}|q^2_{obs,k} - q^2_{cal,k} + \epsilon_{q^2}| \approx q^2_{obs,k}\epsilon_{q^2}$ when $q^2_{cal,k} \approx q^2_{obs,k}$, this likelihood simplifies to

$= \Sigma_{k=1}^{n_k}\ln\left[\sqrt{2\pi q^2_{obs,k}\epsilon_{q^2}}\right]$,

As implemented, the number $\epsilon_{q^2} = 10^{-10}$ is a small value used to prevent a division by zero error. When

Subsampling In the subsampling case, instead of using all 20 peaks, the optimization is performed using only 15 peaks. This is the currently implemented method.

Instead of purely random, subsampling can be based on assignment probabilities. Instead of assigning Miller sets based on the closest reference peak, the assignment neural network is applied to generate a probability distribution over Miller sets for each peak. The most probable peak is choosen and its softmax value is retained. Subsampling is performed by choosing the peaks to be retained with probabilities according to the assignment softmaxes. The softmaxes are normalized so their sum adds to one.

Resampling In the resampling case, all 20 peaks are used, however, the Miler sets are randomly assigned according to their softmaxes.

And of course, it is possible to combined the resampling and subsampling cases.

2.4.4 Redistribution and exhaustive search

This is based on a cartesian distance of unit cell axis lengths. I intend on changing this to NCDIST in the future

To ensure the random optimization finds a correct unit cell, an exhaustive search is implemented. The idea is that some number of candidate unit cells must start the optimization procedure within some distance of the true unit cell to ensure the true unit cell is found. In the case of an entry where the unit cell generator models happens to perform at low efficiency, not enough unit cells will start the optimization in this region to ensure the true unit cell can be found. However, if the unit cell generation models are robust, at least one candidate will be generated within this region. During optimization, after a set number of iterations, the exhaustive search feature redistributes the candidate unit cells, so candidates from densely sampled regions are randomly placed in undersampled regions. This mechanism is also used at the very start of optimization to move candidates from oversampled to undersampled regions.

The initial redistribution and exhaustive search features rely on two parameters, the size of a neighborhood and the maximum number of nearest neighbors. For example, if it is empirically determined that 10 candidate unit cells must start within a 1 Å radius of the true unit cell to ensure it can be found, the maximum number of nearest neighbors is set to 10 and the size of the neighborhood is set to 1 Å.

The initial redistribution algorithm follows the steps:

1. Pairwise differences of all the unit cell axes lengths of each initial candidate are calculated and the number neighbors is calculated for each candidate. This is the number of other candidates within the neighborhood size, 1 Å according to the previous example.

2. The candidate with most neighbors is selected. Its neighboring candidates are randomly selected and removed such that the primary candidate will be left with the maximum number of nearest neighbors.

3. Candidates with less than the maximum number of nearest neighbors are randomly selected and additional candidates are randomly generated by perturbing these selected candidates.

This algorithm is repeated until no candidates have more than the maximum number of nearest neighbors. The exhaustive search mechanism follows this a very similar approach. After a set number of iterations (60 for example), all the candidates receive a small perturbation. Then these candidates are redistributed using the initial redistribution algorithm, except the pairwise differences are calculated between the current candidates and all previous starting candidates.

References

[1] Andrews, L. C., & Bernstein, H. J. (1988) Acta Cryst. A44 1009-1018.

[2] Andrews, L. C., Bernstein, H. J. & Sauter, N. K. (2019) Acta Cryst A75, 593-599.

[3] Andrews, L. C., Bernstein, H. J. & Sauter, N. K. (2019) Acta Cryst A75, 115-120.

[4] Bergmann, J., Le Bail, A., Shirley, R., Zlokazov, V. (2004) Z. Kristallogr. 219 783–790.

[5] Buerger, M. J., (1957) Zeitschrift für Kristallographie 109, 42-60.

[6] Coelho, A. A., (2003) J. Appl. Cryst. 36, 86-95.

[7] Detlefsen, N. S., Jørgensen, M., Hauberg, S. (2019) 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

[8] Habershon, S., Cheung, E. Y., Harris, K. D. M., and Johnston, R. L. (2004) J. Phys. Chem. A. 108, 711-716.

[9] Hull, A. W. & Davey, W. P. (1921) Phys. Rev. 17, 549-570.

[10] Ito, T. (1949) Nature, 164, 755-756.

[11] Lafuente B, Downs R T, Yang H, Stone N (2015) The power of databases: the RRUFF project. In: Highlights in Mineralogical Crystallography, T Armbruster and R M Danisi, eds. Berlin, Germany, W. De Gruyter, pp 1-30

[12] Nix, D. A., & Weigend, A. S., (1994) IEEE,

[13] Oishi-Tomiyasu, R. (2016) Acta Cryst. A72, 73-80.

[14] Rhodes, B., & Gutmann, M. (2022) Enhanced gradient-based MCMC in discrete spaces

[15] Runge, C. V. (1917) Phys. Z. 18, 509-515.

[16] Schriber, E. A. ... (2022) Nature 601, 360-365.

[17] Seitzer, M., (2022) arXiv:2203.09168v2

[18] Shirley, R. (2003) IUCr Comput. Commiss. Newslett. 2, 48–54.

[19] Stirn, A., Knowles, D. A., (2021) arXiv:2006.04910v3

[20] Stirn, A. etal (2022) arXiv:2212.09184v1

[21] Visser, J. W. (1969) J. Appl. Cryst. 2, 89-95.

[22] Wolff, P. M. DE, (1957) Acta. Cryst. 10, 590-595.

[23] Wolff, P. M. DE, (1958) Acta. Cryst. 11, 664-665.

[24] Wolff, P. M. DE, (1961) Acta Cryst. 14, 579-582.

[25] Wolff, P. M. DE, (1968). J. Appl. Cryst. 1, 108-113.