Learning to Predict Material Structure from Neutron Scattering Data

Cristina Garcia-Cardona[a], Ramakrishnan Kannan[b], Travis Johnston[b], Thomas Proffen[b], Katharine Page[b], Sudip K. Seal[b]
[a]Los Alamos National Laboratory, [b]Oak Ridge National Laboratory
.

***Abstract*—Understanding structural properties of materials and how they relate to its atomic structure, while extremely challenging, is a key scientific quest that has dominated the landscape of materials research for decades. Neutron and X-ray scattering is a state-of-the-art method to investigate material structure on the atomic scale. Traditional methods of processing neutron scattering data to decipher the structure of target materials have relied on computing scattering patterns using physics-based forward models and comparing them with experimentally gathered scattering profiles within a computationally expensive optimization loop. Here, we report an initial design of a data-driven machine learning pipeline for material structure prediction that is computationally faster (once trained) and potentially more accurate. We describe the architecture of the ML pipeline and a preliminary benchmarking study of shallow machine learning models in terms of their prediction accuracy and limitations. We show that material structure prediction from neutron scattering data using shallow learning models is feasible to within 90% prediction accuracy for certain classes of materials but deeper models are required for more general material structure predictions.**

I. Introduction

We are witnessing an unprecedented pace in the adoption of machine learning (ML) methods within a wide variety of industrial and technological domains. Natural language processing, autonomous vehicles, image analysis and facial recognition are some of the often cited application areas in which ML has been demonstrated to achieve impressive advances over more traditional methods. ML-based approaches are generally used to model solutions to problems for which no closed form analytical understanding exists. The predictive capabilities of many ML models have been found to be competitive and, on numerous occasions, superior to existing state-of-the-art application-specific methodologies.

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan) A portion of this research used resources at the Spallation Neutron Source, a DOE Office of Science User Facility operated by the Oak Ridge National Laboratory. A portion of this research used resources at the ALCF, a DOE Office of Science User Facility operated by the Argonne National Laboratory. This research was sponsored by the Exascale Computing Project, DOE.

While many of the fundamental concepts of ML (and artificial intelligence) can be traced back several decades, the data deluge of recent years has enabled an explosive re-emergence of ML as a dominant methodology in big data analytics. This is because ML models often require (a) a substantial amount of data to train on, and (b) large computing resources, both of which are increasingly becoming available today. Despite a veritable revolution in big data analytics, adoption of ML for scientific knowledge discovery at-large has lagged behind their commercial counterparts such as information retrieval, image processing and voice assisted services. One amongst several reasons for this delay can be attributed to the fact that the size, availability and readiness of data for ML models to train on differ significantly from one scientific domain to another. But, new scientific domains are constantly testing the advantages and limitations of ML methodologies for knowledge discovery. The work reported here embodies one of the first large-scale attempts to use ML approaches for material structure prediction using neutron scattering data.

A. Motivation

Understanding the atomic structure of materials enables scientists to unravel the structural, mechanical and thermal properties of the material and help them design new materials or improve existing ones. Neutron and X-ray diffraction allow researchers to unravel the atomic structure of complex materials and provided crucial scientific insight into our current understanding of many materials from superconductors to batteries to protein structures. While neutron and X-ray diffraction provide complementary information about each sample, the crystallographic analysis methods of the data collected by these techniques are very similar. As such, ML methods developed for the analysis of one kind of data are applicable for the analysis of the other kind with minimal modifications.

For the purpose of this paper, we limit the scope of diffraction analysis to obtaining the average atomic structure from Bragg scattering of neutrons. Structure determination and refinement are largely manual processes that require extensive expert input from scientists making this a bottleneck towards automatic and fast structure determination from diffraction data. This bottleneck is becoming more significant as experimental instrumentation continues to improve, producing more data more rapidly at state-of-the-art

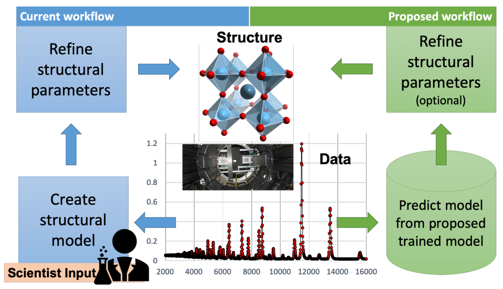

Current workflow
Proposed workflow
Refine structural parameters
Structure
Refine structural parameters (optional)
Data
Create structural model
Predict model from proposed trained model
Scientist Input

Figure 1: Typical workflow in structure determination and refinement. The loop on the left show the current workflow involving a domain expert creating a structural model. The loop on the right is the workflow described here where the structural model and parameter estimated are obtained from a trained network.

facilities such as the Spallation Neutron Source (SNS) in the Oak Ridge National Laboratory.

The current workflow for material structure determination from neutron scattering data is shown on the left in Fig. 1. In it, a suitable structural model is built by a researcher through (educated) trial and error which is typically very time-consuming and quickly becomes a bottleneck in scientific discovery. The proposed workflow, shown by the right loop in Fig. 1, is expected to replace the 'manual' model building by interrogating a trained network, as described in this paper. This approach will ultimately predict likely structural models with structural parameter estimates. Accordingly, the primary motivation of the effort reported here is to build an alternative fast and potentially more accurate data-driven ML approach for the inverse problem of material structure determination from neutron scattering data.

B. Related Work and Contributions

ML for neutron scattering data analysis is a nascent area of research. Earlier this year, the use of principal component analysis with an artificial neural network to predict neutron scattering cross-sections to constrain the parameters of a pre-existing model Hamiltonian was reported in [1] while [2] reports an unsupervised ML approach to studying phase transitions in single crystal x-ray diffraction data. An ML-based approach to classify the local chemical environment of specific metal families from the simulated K-edge XANES of a large number of compounds was reported in [3]. To the best of our knowledge, no effort to use ML techniques to predict material structure from neutron scattering data has been reported till date.

The target class of materials used in this study is what is called a *perovskite* compound whose general molecular signature is represented by ABO_3 where A and B refer to general chemical elements but O always represents oxygen. In particular, we use the perovskite *barium titanate*, $BaTiO_3$, as our target material. Perovskite materials have gained world-wode interest in recent years for a variety of reasons and advances made in faithful perovskite material structure predcition are expected to be highly impactful to

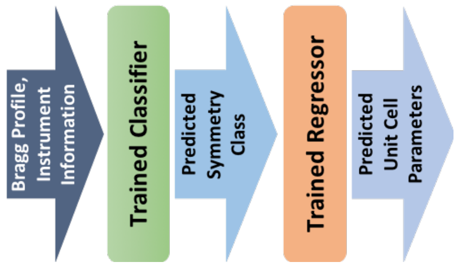

Bragg Profile, Instrument Information
Trained Classifier
Predicted Symmetry Class
Trained Regressor
Predicted Unit Cell Parameters

Figure 2: The proposed ML pipeline.

the global materials science community. It is worth noting that the overall ML framework presented here is not specific to perovskites and is equally applicable to any material sample. Specifically, in this paper, we present:

- the first reported large-scale generation of labelled sets of neutron scattering data for machine learning models to train on.
- the design and benchmarking of one of the first classifiers for crystallographic symmetry group prediction.
- the first application of a number of popular regression models to demonstrate their efficacy and limitations in predicting the structural parameters of (perovskite) materials.

Together, these results represent one of the first attempts to predict material structure directly from their neutron scattering profiles.

II. Preliminaries

Crystalline materials are characterized by translational periodicity of the basic unit called the unit cell. Three-dimensional space allows for arrangements falling into seven crystal classes and 14 Bravais lattices [5] shown in Fig.

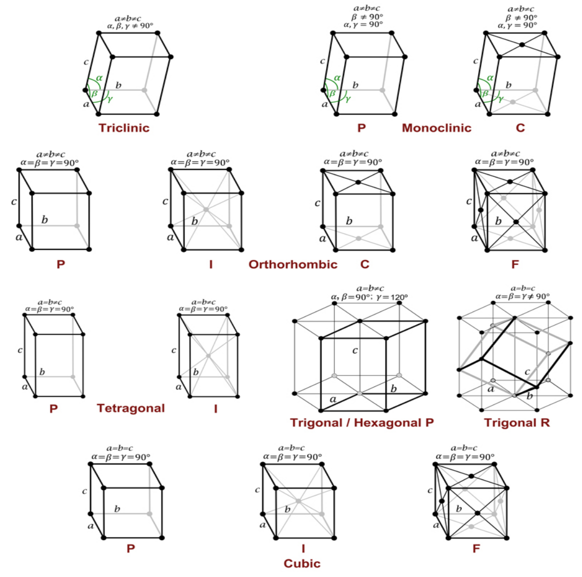


Figure 3: The 14 Bravais lattices and 7 crystal classes that are compatible with three-dimensional translational periodicity. [4]

3. These seven crystal classes are called cubic, tetragonal, orthorhombic, hexagonal, trigonal, monoclinic and triclinic. Each of these classes are characterized by specific constraints on the lattice parameters a, b, c defining the length of the unit cell in each dimension and the unit cell angles α, β and γ. Examples of these constraints and test data generated are shown for the cubic, trigonal and tetragonal crystal classes in Table I. Accordingly, the structure determination of any material is a two step process wherein the Bravais lattice and ultimately the space group is determined in the first step. The second step is to determine the unit cell lengths a, b, c and unit cell angles α, β, γ. The final goal is the determination of the unit cell content (e.g. atomic coordinates, atomic displacement parameters) allowing the researcher to build a complete model of the crystal structure. The proposed pipeline that accomplishes these two steps is shown in Fig. 2. The input consists of a Bragg profile and a description of the instrument that collected the diffraction data. This information is used by a classifier to predict its crystallographic class (one of seven). The predicted class, along with the Bragg diffraction pattern, are then used by a regressor to predict the lattice parameters and bond angles of the material.

A major challenge for training classifiers and regressors for use in the preceding pipeline is a lack of *labelled* neutron diffraction data for these ML models to train on. This challenge is addressed by using a forward surrogate physics model with low computational cost to simulate large data sets that sample the complete parameter space of the inter-atomic distances and bond angles subject to the crystallographic class relations as exhaustively as possible. The ML models are then trained on a large subset of these simulated samples, tested on a held out subset and then validated against experimentally annotated diffraction data obtained from NOMAD, the Nanoscale Ordered Materials Diffractometer at the Spallation Neutron Source, Oak Ridge National Laboratory [6]. Generation of this training data set is described next.

III. Data Generation

A. Generation of Training Data

The training data set is generated using the GSAS-II software tool [7]. GSAS, which stands for Generalized Structure Analysis System, is a software that is globally used to determine structures of crystalline solids at all length scales and compositions (powder or single crystal) when probed with either neutrons or X-ray. Here, GSAS-II is used to compute the diffraction pattern, X, for every combination of lattice parameters, Y. Here, Y denotes a set $\{a, b, c, \alpha, \beta, \gamma\}$ of lattice parameters and unit cell angles as defined in Section II. GSAS-II requires an instrument parameter file to model the physics of the diffractometer used to generate the diffraction profile. As such, a parameter specification file corresponding to the NOMAD instrument is used for the GSAS-II tool to maintain consistency with the NOMAD-generated experimental data against which the model predictions are eventually validated.

Table I: GSAS-II Diffraction Pattern Simulation.

Class	Parameters	Simulations (n)	Size
Cubic (predict a)	$a = b = c$ $\alpha = \beta = \gamma = 90°$	3,000	0.13 GB
Trigonal (predict a, α)	$a = b = c$ $\alpha = \beta = \gamma \neq 90°$	579,000	25 GB
Tetragonal (predict a, c)	$a = b \neq c$ $\alpha = \beta = \gamma = 90°$	998,584	42 GB

Each diffraction pattern X is a set of 2807 2-tuples $(y,\ I(y))$ where y is the time-of-flight (ToF), simulated in the range [1,360μs, 18,919μs], and $I(y)$ is the corresponding intensity. As mentioned earlier, the fast GSAS simulations generate the training samples. These fast simulations enable high resolution sampling of the total parameter space spanned by the lattice parameters and unit cell angles. Simulation of each diffraction pattern typically takes between 2s to 17s, depending on the particular combination of crystallographic symmetry and structural parameters.

Accordingly, labelled training data sets for each crystallographic class were generated by sweeping the space of lattice parameters and unit cell angles with step resolution of 0.001 for the lengths and $0.5°$ for the angles. For cubic class, a was simulated in the range [2.5, 5.5]. For trigonal class, a was simulated in the range [2.5, 5.5], while α was simulated in the ranges [20°, 88°] and [92°, 120°]. For tetragonal class, a, c were simulated in the range [3.5, 4.5]. The GSAS-II software is a sequential tool. To enable large-scale generation of labelled training data with maximum possible coverage of the total parameter space across all seven crystallographic classes, an MPI-based parallel framework was developed to efficiently generate the diffraction patterns in a concurrent and distributed manner. MPI4Py [8] was used to implement the distributed framework, with which the parameter grid is partitioned across a pool of processors after which each processor locally executes GSAS-II to simulate the diffraction pattern for every locally owned tuple, and carried out independently for each of the crystallographic symmetry classes. Table I summarizes the number of simulations performed and the size of data generated.

B. Experimental Data Annotation

Experimental neutron powder diffraction data of $BaTiO_3$ as a function of temperature was collected on the NOMAD instrument housed in the Spallation Neutron Source at Oak Ridge National Laboratory. The sample temperature of the neutron diffraction experiments ranged from $T = 100K$ to $T = 460K$ and covers phases corresponding to crystal systems from trigonal $\rightarrow$ orthorhombic $\rightarrow$ tetragonal $\rightarrow$ cubic respectively. In all cases, 'traditional' structure analysis (see Fig. 1) was carried out to obtain the structural parameters.

Table II: Notations

Notation	Description
n	Number of training data samples for each crystallographic class
N	Total number of samples across all crystallographic classes
d	Number of features or length of $I(q)$
$l \in \mathbb{R}_+$	Number of distinct prediction parameters for a crystallographic class (1 for cubic system, 6 for triclinic, etc.)
$X \in \mathbb{R}_+^{n \times d}$	Training data
$Y \in \mathbb{R}_+^{n \times l}$	Regression/Classification labels for data X
$W \in \mathbb{R}^{d \times l}$	Non-negative solution weight matrix from NNLS
$\|A\|_F^2$	Frobenius norm, defined as $\sum_{ij} a_{ij}^2$

The crystallographic class and the lattice parameter set $\{a, b, c, \alpha, \beta, \gamma\}$ were used as labels.

IV. Machine Learning Framework

As mentioned earlier, the ML pipeline (see Fig. 2) presented here has two main components, namely, (a) a classifier for the prediction of crystallographic classes, and (b) a regressor to predict the unit cell parameters (lengths and angles) for the symmetry class predicted by the classifier.

Let a classifier and a regressor be denoted by the functions $f_1 : X \to Y$ and $f_2 : X \to Y$, respectively. Here, $X \in \mathbb{R}_+^{n \times d}$, n is the number of samples, d is the number of features, $Y \in \mathbb{R}_+^{n \times l}$. We define a label $Y \in \mathbb{R}_+^{l}$, where $l \in [1, 6]$ is the number of labels to be predicted by the regressor. Note that the value of l varies with the symmetry class predicted by the classfier. For example, $l = 1$ for a cubic crystallographic class as the unit cell length a completely defines it. Similarly, $l = 6$ for a triclinic crystallographic class, all the six different parameters a, b, c, α, β and γ are required. The objective, here, is to learn the functions f_1 and f_2.

Two loss functions are defined. The training loss is defined as the L_2 norm between the estimated parameter set $\|\hat{Y} - Y\|_2^2$ while the inference loss is defined as $\hat{x} - x_{test}$. Here, x_{test} is a test sample and $f_2 : x_{test} \to \hat{y}$. In other words, $\hat{y}$ is the label predicted by the model f_2 on the test data x_{test}. The diffraction pattern, $\hat{x}$, is computed by the forward simulation using the predicted $\hat{y}$ as its input.

A. Classifier for Crystallographic Symmetry

Here, we describe an initial prototype of a classifier trained on simulated data, as described in Section III. This preliminary design is shown to be capable of distinguishing five of the seven total crystallographic symmetries, namely, cubic, tetragonal, trigonal, monoclinic, and triclinic. To understand the relationship between the several classes we apply multidimensional scaling (MDS) to random subsets of 500 examples from each class. MDS is a method of dimensionality reduction; in this case, we transform each $I(Q)$ vector (length 2,807, one vector for each of the 2,500 random examples) into a 2-dimensional vector. The MDS

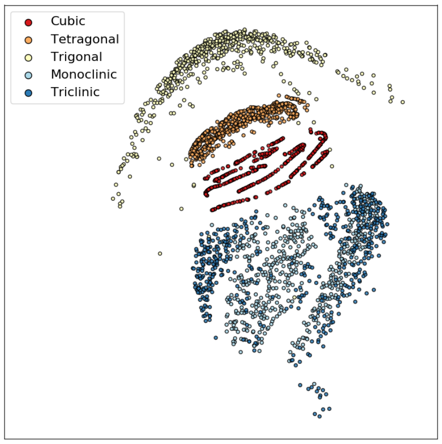


Figure 4: The result of multi-dimensional scaling applied to random subsets of 500 examples from each class.

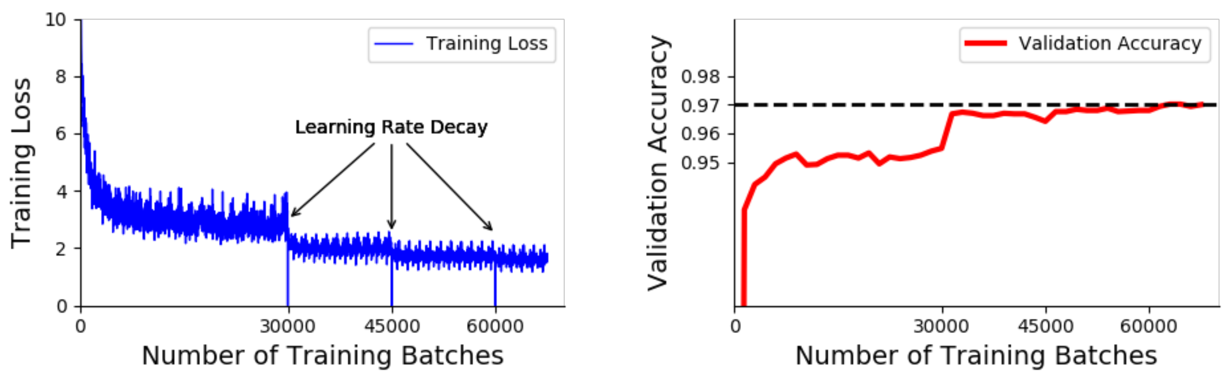
process attempts to find lower-dimensional vectors such that pairwise distances are preserved as much as possible, i.e. if $\vec{a}_i$ and $\vec{a}_j$ are two vectors in $\mathbb{R}^{2807}$ and $\vec{b}_i$ and $\vec{b}_j$ are corresponding vectors in $\mathbb{R}^2$, then $\|\vec{a}_i - \vec{a}_j\| \approx \|\vec{b}_i - \vec{b}_j\|$. Figure 4 shows the results of the MDS; class labels are indicated by color. While the specific *shape* resulting from MDS is irrelevant, it does represent useful information. For example, well-separated clusters in the MDS map imply that distances are approximately preserved. As such, one can conclude that simple distance-based learning algorithms–such as k-nearest neighbors–would easily distinguish the cubic, tetragonal, and trigonal classes. However, considerable overlap between the monoclinic and triclinic classes suggests a poor likelihood of any similar distance-based algorithm to be effective at differentiating the two classes.

The learning rate was manually reduced at several points in training (after 30,000, 45,000, and 60,000 batches trained). The model was found to converge after training on 75,000 batches of data. Overall, the network was demonstrated to be very successful at differentiating cubic, tetragonal and trigonal classes in the simulated data with a validation accuracy that reached as high as 97%.

The observation that distance-based metrics are unlikely to work motivated the design and use of a simple, convolutional neural network (CNN) to learn the distinction. A simple CNN which applies 1D convolutions directly to the $I(Q)$ vector was implemented as follows:

- 1D Convolution (16 learned filters, kernel width=3, stride=1)
- 1D Max Pooling (kernel width=2, stride=2)
- 1D Convolution (32 learned filters, kernel width=4, stride=2)
- 1D Max Pooling (kernel width=2, stride=2)
- Fully Connected (256 hidden neurons, w/ReLU activation)
- Fully Connected (5 output neurons, w/Softmax)

This CNN was trained on 10,000 random examples (2,000



(a) (b)

Figure 5: (a) Neural network training on 2,000 examples per class validating on a disjoint subset of 1,000 examples per class. (b) Accuracy on validation set is top-1.

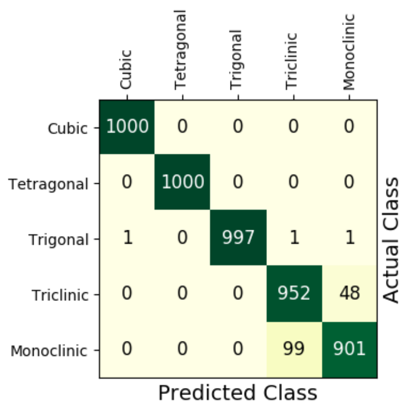


Figure 6: Confusion matrix of the trained neural network on the validation set. The neural network has the most trouble distinguishing monoclinic from triclinic examples achieving only 92.65% accuracy on these classes. It achieves near 100% accuracy on the other classes.

per class) and validated on a reserved set of 1,000 examples per class (5,000 total examples). The loss function was cross entropy loss and the network was trained using stochastic gradient descent (SGD).

Figure 5a shows the training loss (left) and the validation accuracy (right) as functions of the number of batches trained. As expected, the CNN-based classifier differentiates the cubic, tetragonal, and trigonal classes (just as a nearest neighbors approach would have been very as well) with near 100% accuracy, only mis-classifying 3 trigonal examples out of 3,000 total. On the triclinic and monoclinic classes, the CNN-based classfier, however, showed significant improvement, successfully predicting with about 92.65% accuracy but still leaving considerable room for improvement at the same time.

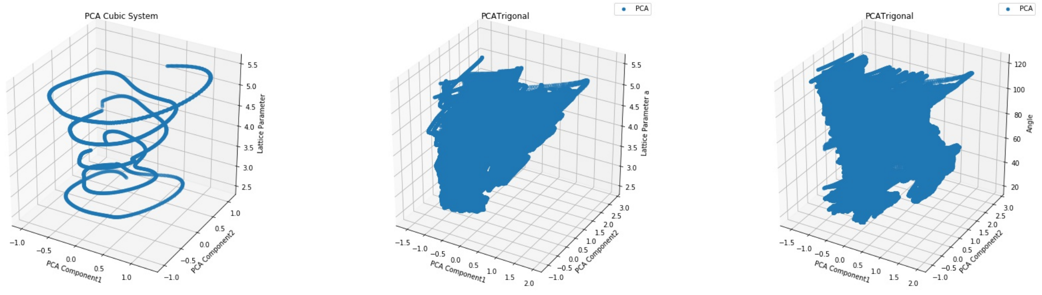
B. *Cell Parameter Regressor*

As outlined in Fig. 1, a regressor is required to predict the unit cell parameters corresponding to the particular symmetry class predicted by the classifier. The purpose of the regression models is to synthesize a mapping from the diffraction pattern to the structural parameters. A different regression model is trained for each of the crystallographic symmetries studied. To better understand this task, different machine learning regression models are first compared to establish a baseline performance to compare potentially more sophisticated deep learning models in future.

For any symmetry class, answers to the following two questions are sought, namely: (a) what is the relationship between the labels Y and the training data X, and (b) what are the relationships amongst the labels? Additionally, we need to ascertain if a multilabel regressor is necessary.

To address the former question, principal component analysis (PCA) results of training matrix X with labels Y for one parametric cubic and two parametric trigonal symmetries are presented in Fig. 7. Similar analyses and observations can be extended up to six parameters triclinic systems without loss of generality. The simplest case of cubic symmetry class needs only one lattice parameter a while the trigonal symmetry class involves two lattice parameters a and α. A regression model f_2 is chosen accordingly. Figure 7 illustrates that relationship between the training data X and the individual labels $y_i \in Y$ is non-linear suggesting that a non-linear ML methods like support vector regression (SVR) is expected to perform better than linear least squares methods.

The covariance between the labels α and a across 579000 samples was found to be extremely small $\left(\sim 10^{-12}\right)$, establishing the fact that these labels are uncorrelated. The lack of covariance is not surprising since the training data was generated by the forward GSAS II model using input parameters (unit cell lengths and angles) that were independently grid-sampled in the space spanned by the (symmetry) class-specific parameters in $\{a, b, c, \alpha, \beta, \gamma\}$. That is, each training sample was generated by fixing the parameters $y_i \in Y - \{y_j\}$, where $j \neq i$, and sampling y_j between its lower and upper bounds by a given step size. As such, the values of y_j are completely independent of the values of y_i.



(a) (b) (c)

Figure 7: (a) Cubic class. (b) Trigonal class with cell length a. (c) Trigonal class with cell angle α.

Thus, the regression models f_2 for every y_i, can not only be independent, but also different. Accordingly, we tested multiple independent models, such as, unconstrained least squares, multi-label regressor with gradient boosted trees and support vector machine (regressor), but only present results from random forest regression (RFR) and non-negative least squares (NNLS) for f_2 across all the labels Y in this paper.

1) Non-negative Least Squares (NNLS): The fundamental baselines to solve the inverse problem f_2 is to determine a weight matrix $W \in \mathbb{R}^{d \times l}$. Assume that given a test experimental point $x_{test} \in \mathbb{R}^d_+$ when multiplied with the weight, W, yields the estimated labels. That is, $\hat{y} = W x_{test}$. Finding the least squares is solving the problem of determining the weight matrix W, given the training data X and labels Y. Additionally, if $W \geq 0$, where every entry $w_{ij} \in W \geq 0$, then the problem is that of computing the non-negative least square (NNLS). Formally,

$$\underset{W \geq 0}{\operatorname{argmin}} \|XW - Y\|_F^2 + \lambda_1 \|W\|_F^2 + \lambda_2 \|W\|_{\ell_1} \qquad (1)$$

Multiple algorithms exist that solve the problem of NNLS. Chen and Plemmons explain these algorithms briefly in [9]. In this paper, an active set variant of the NNLS problem, called block principal pivoting [10] by Kim and Park, is adopted. The ℓ_2 regularization on W tames the growth of the values and ℓ_1 regularization makes it insensitive to outliers.

2) Random Forest Regression (RFR): Random forest (RF) is a type of ensemble predictors that can solve classification or regression problems [11]. The goal behind ensemble methods is to train a group of individual (perhaps weak) predictors and aggregate the individual results in order to generate better predictions than any of the individual models. In general, the aggregation of results yields predictions with a bias similar to those of individual models but with a lower variance, which improves the robustness of the ensemble model and contributes to a better generalization over data not seen during training.

A random forest, in particular, is an aggregation of decision trees. Specifically, the Scikit-Learn python package [12] used to train the random forest regression models, builds binary decision trees. A decision tree is a construct that allows to compute predictions by traversing the nodes of the tree from root to a terminal, childless, node (i.e. leaf). Each node j in a binary tree, that is not a leaf, has two children and is characterized by a feature and a threshold (f_j, t_{f_j}). For each sample $\boldsymbol{x_i}$ in the data set it is possible to traverse the binary decision tree until a leaf node that includes a prediction (class or continuous regression value) is reached. To traverse a node, the value of the f_j feature in the sample point $\boldsymbol{x_i}$ is compared to the node's threshold t_{f_j} and the path continues towards left or right child, depending on the result of the comparison (i.e. $\leq$ or $>$ than the threshold). The prediction computed in each leaf corresponds to the consolidation of the output values of all the training samples that land on that specific leaf. In general, the consolidation corresponds to the most abundant class in the node (classification), or the mean value of the output of the training samples associated with it (regression).

To train a decision tree, i.e. building the tree, Scikit-Learn uses the Classification and Regression Tree (CART) algorithm [13]. The CART algorithm for regression splits the training set in two subsets using a single feature f_j and a threshold t_{f_j}. The threshold and feature are selected by finding the (f_j, t_{f_j}) pair that yields the lowest mean squared error (MSE) (2) weighted by the sizes of the left and right subsets. The splitting proceeds recursively for each of the branches of the previous split, until a pre-defined depth is reached or no further partition reduces the MSE. Note that this is a greedy, not necessarily optimal, algorithm. Note also, that in order to keep the computation of the tree tractable, Scikit-Learn randomly selects the set of features to evaluate at each node [12], [13].

Ensemble methods exhibit better performance when the individual predictors have low correlation with each other [11]. One alternative to achieve this is to train the individual models using different random subsets of the training data. Scikit-Learn generally builds a RF model by bagging, i.e. building the group of decision trees by sampling random subsets with replacement. Randomly selecting the features to evaluate at each node, instead of searching for the very best splitting features, not only contributes to a faster training, but it also produces greater tree diversity [13]. In order to control the complexity of the RF and prevent the model from overfitting, it is common to control the growth of the trees (e.g. limit the depth of the individual trees or set a minimum number of sample points per leaf) and control the ensemble itself (e.g. number of individual trees in the forest).

V. Performance

A. Computational Metrics

The performance for the regression models was compared in terms of the mean squared error (MSE)

$$\mathrm{MSE}(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \ , \tag{2}$$

where n is the total of samples, $\hat{y}_i$ is the predicted value of the i-th sample and y_i the corresponding true value. The other performance measure used was the coefficient of determination (R^2), as computed by the scikit-learn python package [12]:

$$R^2(y,\hat{y}) = 1-\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \ , \quad \bar{y}=\frac{1}{n}\sum_{i=1}^{n}y_i \ . \tag{3}$$

This coefficient provides an indication of goodness of fit with a best possible score of 1.0. Since the error measures are unnormalized, predictions from RFR and NNLS trained models can be different.

B. Experimental Data

The phase diagram of BaTiO$_3$ establishes a correspondence between its crystallographic classes and the temperature (T). Briefly, this correspondence is as follows. $T < 200K$ corresponds to trigonal, $200K \leq T < 270K$ is orthorhombic, $270K \leq T < 400K$ is tetragonal and $T \geq 400K$ is cubic. In all, the experimental data used to validate the proposed prediction pipeline includes 15 samples. These experimental samples were measured at the following temperatures: 100K, 300K, 325K, 340K, 355K, 370K, 385K, 390K, 395K, 400K, 405K, 415K, 430K, 445K and 460K. Accordingly, our experimental data includes: 1 trigonal, 8 tetragonal and 6 cubic samples of $BaTiO_3$.

The experimental data was preprocessed to subtract the background (since our simulations are carried out without background) and to match the x-coordinates that are slightly different between the two. The background is modelled using a 12-degree Chebyshev polynomial of the first kind, and the diffraction patterns are aligned using the closest x-coordinate from the simulation. The background adjustment is sensitive to the peaks in the diffraction pattern. Thus a maximum clip value is used to reduce the peak influence.

C. Quality of Predictions

RFR models were constructed for three crystal classes: trigonal, tetragonal and cubic, for which experimental measurements were available, as described in the preceding section. A different regression model was trained for each of these crystal classes. The input features correspond to the vector of y-coordinates of each diffraction pattern, i.e. 2,807 features (the vector of x-coordinates is not used since it is the same for all the simulated diffraction patterns). The values of different sets of lattice parameters are predicted for each crystal class (see Table I). For trigonal and tetragonal classes, the RFR models were trained using a 5-fold cross validation (CV) scheme, over a random selection of 50% of the data. Due to the small size of the cubic dataset, a 10-fold CV scheme was used over all the cubic data instead and 100 trees with maximum depth of 10 and a minimum of 30 samples per leaf were constructed. For the trigonal and the tetragonal data, 200 trees with maximum depth of 15, and a minimum of 30 samples per leaf, were constructed. No optimal hyper-parameter search was made for the algorithm. Instead, parameters were manually set to obtain good performance. The performance was compared in terms of the mean squared error (MSE) and the coefficient of determination (R^2). A summary of mean and standard deviations for MSE and R^2 from the CV runs is reported in Table III and IV using random forest and NNLS respectively. Note that these are evaluated on the subset of the fold not used for training.

From these tables, we can observe that random forest outperformed NNLS. This is because, NNLS is good in capturing linear relationships. However, Figure 7 showed that the relationships between X and Y are nonlinear.

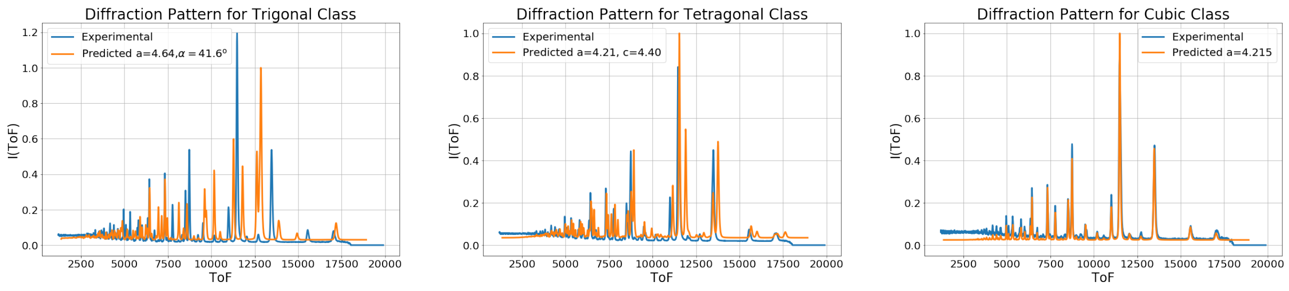
Table III: Random forest regression on simulated data.

Class	**MSE**	$\boldsymbol{R^2}$
Trigonal	$2.52\times10^{-3} \pm 1.74\times10^{-4}$	$0.995 \pm 3.15\times10^{-4}$
Tetragonal	$1.57\times10^{-5} \pm 1.58\times10^{-6}$	$0.999 \pm 1.92\times10^{-5}$
Cubic	$1.20\times10^{-3} \pm 6.53\times10^{-4}$	$0.998 \pm 8.02\times10^{-4}$

Table IV: NNLS on simulated data.

Class	**MSE**	$\boldsymbol{R^2}$
Trigonal	$5.44\times10^{2} \pm 3.71$	$-0.117 \pm 3.96\times10^{-3}$
Tetragonal	$8.16\pm1\times10^{-2}$	-96.85 ± 0.17
Cubic	$4.13\times10^{-3} \pm 2.04\times10^{-3}$	$0.9943 \pm 2.91\times10^{-3}$

Since RFR was shown to outperform NNLS on simulation data, only the MSE performance for the diffraction patterns generated with the RFR predicted values vs. the experimental data is presented in Table V. Figure 8a includes a comparison between the experimental trigonal measurement and the diffraction pattern of the RFR predicted parameters:



(a) (b) (c)

Figure 8: Diffraction pattern of experimental vs. RFR predicted values for (a) trigonal sample, (b) best matched tetragonal sample and (c) best matched cubic sample. Time-of-Flight (ToF) in μs vs. Intensity, I(ToF).

$a = 4.64$ and $\alpha = 41.6°$, which has MSE = $9.52{\times}10^{-3}$. The best tetragonal match is shown in Figure 8b, for RFR predicted parameters of $a = 4.21$ and $c = 4.40$, with MSE = $3.68{\times}10^{-3}$. The best cubic match is plotted in Figure 8c, for RFR predicted parameter of $a = 4.215$, with MSE = $9.53{\times}10^{-4}$.

Table V: RFR results on experimental data.

Class	**MSE**
Trigonal	$9.52{\times}10^{-3}$
Tetragonal	$7.38{\times}10^{-3} \pm 1.4 \times 10^{-3}$
Cubic	$6.63{\times}10^{-3} \pm 2.56 \times 10^{-3}$

VI. Discussion and Future Work

Results for RFR show that it is possible to get good lattice parameter predictions for the experimental measurements of the cubic crystal class. In contrast, despite having good performance on the simulated datasets, the lattice parameter prediction for the experimental measurements of the tetragonal and trigonal crystal classes is not as good. The regression for the parameters of the tetragonal class seems to generate diffraction patterns that capture most of the peaks, having a good agreement in intensity and slight mismatches in position. The regression for the parameters of the trigonal class generate diffraction patterns that capture some of the peaks, but with appreciable mismatches in peak position and intensity. Currently, the shallow models use a relatively simple set of features that do not take into account the spatial distribution of peaks explicitly. While this seems to be enough for simulated data, our results suggest that more sophisticated models are required to handle experimental data. Moreover, to apply the machine learning models to experimental data, it is necessary to deal with practical issues that do not appear that crucial when using simulated data. In particular, there is an undesirable sensitivity to the background fitting procedure that should be improved in the future, in order to guarantee more robust predictions.

This paper reports the preliminary findings of an ongoing first-of-its-kind effort to directly predict material structure from neutron scattering data using trained machine learned models. The results here not only establish the baseline performance of shallow machine learning models, but also exposes their limitations in materials structure prediction opening up the clear need to test and explore deep learning methods.

References

[1] R. Twyman, S. Gibson, J. Molony, and J. Quintanilla, "A machine learning approach to magnetic neutron scattering," March 2019.

[2] J. Venderley, M. Matty, and E.-A. Kim, "Unsupervised machine learning of single crystal x-ray diffraction data," March 2019.

[3] D. Lu, M. Carbone, M. Topsakal, and S. Yoo, "Using machine learning to predict local chemical environments from x-ray absorption spectra," March 2019.

[4] M. Martinez-Ripoll. (2014) Crystallography. Departmento de Cristalografía y Biología Estructural. [Online]. Available: http://www.xtal.iqfr.csic.es/Cristalografia/

[5] C. Kittel, *Introduction to Solid State Physics*, 8th ed. Wiley, 2004.

[6] J. Neuefeind, M. Feygenson, J. Carruth, R. Hoffmann, and K. Chipley, "The nanoscale ordered materials diffractometer nomad at the spallation neutron source sns," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 287, pp. 68–75, 2012.

[7] B. H. Toby and R. B. Von Dreele, "GSAS-II: the genesis of a modern open-source all purpose crystallography software package," *Journal of Applied Crystallography*, vol. 46, no. 2, pp. 544–549, 2013.

[8] L. Dalcin, "MPI for Python," 2019. [Online]. Available: https://mpi4py.readthedocs.io/en/stable/

[9] D. Chen and R. J. Plemmons, "Nonnegativity constraints in numerical analysis," in *The birth of numerical analysis*. World Scientific, 2010, pp. 109–139.

[10] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[11] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[12] "scikit-learn: Machine Learning in Python," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score

[13] A. Géron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow*, 1st ed. O'Reilly, 2017.