Insightful Classification of Crystal Structures using Deep Learning

CH5650: Molecular Data Science and Informatics Term Project

Vir Karan (MM19B057)

1 Introduction

Characterization of any material begins with determining its composition and crystal structure, i.e., what atoms are present in the substance and how they are arranged in space (on a lattice). The first step in predicting the material's properties is to have this information. Finding the group of all transformations under which the system is invariant is an effective technique of classifying crystals, and in three dimensions, these transformations are described by the idea of space groups. To find the space group of a particular structure, one must first discover the permissible symmetry operations, which must then be compared to all conceivable space groups in order to acquire the correct label. This method works for perfect systems, but real-world data contains defects, impurities and noise. To deal with this, thresholds must be established to determine how loose one wishes to be while classifying. Then there's the problem of varying thresholds resulting in different classifications. Furthermore, any method for automatically classifying crystals based solely on symmetry operations is unlikely to work if the crystal is rotated or has defects, as this would break the symmetry elements with respect to the axis system studied.

Hence this work, originally titled "Insightful Classification of Crystal Structures using Deep Learning" by Angelo Ziletti, Devinder Kumar, Matthias Scheffler & Luca M. Ghiringhelli [5] has been considered and reproduced in this report. In this work, the authors: (1) Introduce an automatic procedure to classify crystal structures starting from a set of atomic coordinates and lattice vectors. (2) Introduce a way to represent crystal structures (by means of images). (3) Present a classification model based on convolutional neural networks (ConvNets or CNNs). (4) Unfold the internal behavior of the classification model through visualization. Furthermore, as discussed in class, through this report this work has been further extended to pose the problem of crystal structure classification using unsupervised learning techniques as well. The code and data used to prepare the results of this report have been uploaded on GitHub[1] for reference as well.

[1] Can be found on: `www.github.com/vir-k01/CH5650`

2 Methods

2.1 2D Diffraction Fingerprint

The initial stage in any machine learning process is to represent the material under consideration in a manner that a computer can understand. This representation should include all relevant system information needed for the desired learning task. However, in order to reflect the infinite nature of crystals, it is critical that the descriptor captures system symmetries in a compact manner while being size invariant. Reciprocal space exhibits periodicity and prevailing symmetries, so switching to reciprocal coordinates is advantageous. The scattering of an incident plane wave through the crystal may be simulated, and the diffraction pattern in the detector plane orthogonal to that incident wave can then be determined. This experiment is schematically shown in Figure 1. The amplitude ψ, which originates from the scattering of a plane wave with wave vector k_0 by N_a atoms of species a at positions $x_j^{(a)}$ in the material can be written as:

$$\Psi(\mathbf{q}) = r^{-1} \sum_a f_a^{\lambda}(\theta) \left[\sum_{j=1}^{N_a} r_0 \exp\left(-\mathrm{i}\mathbf{q} \cdot \mathbf{x}_j^{(a)}\right) \right] \tag{1}$$

Here, r_0 is the Thomson scattering length, $q = k_1$ - k_0 is the scattering wave vector, x the corresponding position in the detector plane, and $r = |x|$. Assuming elastic scattering, we have that $|k0| = |k1| = 2\pi/\lambda$, where λ is the wavelength of the incident radiation. The quantity $f_a^{\lambda}(\theta)$ is the X-ray form factor, it describes how an isolated atom of species a scatters incident radiation with wavelength λ and scattering angle θ. The intensity of the diffraction pattern formed on the detector plane is then given by:

$$I(\mathbf{q}) = A \cdot \Omega(\theta) |\Psi(\mathbf{q})|^2 \tag{2}$$

where $\Omega(\theta)$ is the solid angle covered by the (theoretical) detector, and A is a (inessential) constant determined by normalization with respect to the brightest peak.

For each structure, first the standard conventional cell is constructed [4]. Then, the structure is rotated 45^o clockwise and counterclockwise about a given crystal axis (say, x axis), and the diffraction pattern for each rotation is calculated and superimposed. After that, the technique is repeated for each of the three crystal axes. The final output is a single RGB image of the crystal structure, with each colour channel displaying the diffraction patterns acquired by rotating around a specific axis. The central spot is removed from the produced patterns since it provides very little information about the symmetry elements present in the crystal and also reduces the contrast for higher order peaks. As a result, each system is defined as an image, which is referred to as the two-dimensional diffraction fingerprint (D_F). The above process can be performed using the functions present in the ai4materials library [2].

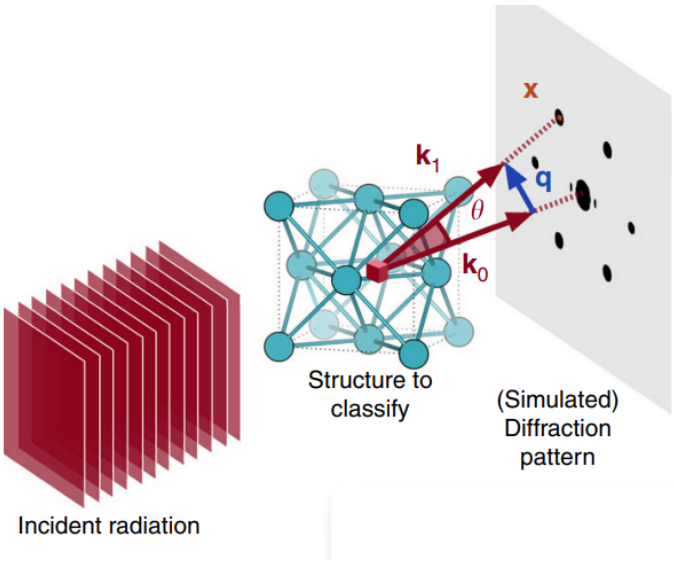

Figure 1: A schematic showing the interaction of an incident plane wave with a sample crystal considered, and the diffraction pattern thus obtained. (Image taken from [5])

2.2 Dataset

Since the aim is to predict the crystal structure given the atomic coordinates and lattice vectors of the crystal, firstly, for each structure in the dataset, three random rotations of the initial structure around the three crystal axes are done and the resulting structures are concatenated to randomize the initial crystal orientation. This way, any model trained would be robust to changes in the initial crystal orientation and change of the axes system used. Then, the standard conventional cell is constructed and replicated in all three directions such that the resulting cluster contains around 250 atoms. Defective structures are then generated from these supercells by randomly removing 25% of the atoms. A total of 10,517 structures are made and used only to train the model, and another 10,517 defected structures are used for testing. A wavelength of $\lambda = 5.0 \times 10^{-12}$ m for the incident plane wave is used, which is a wavelength typically used in electron diffraction experiments. Some example structures for each class alongwith their corresponding diffraction fingerprint are shown in Figure 2A and B respectively. As can be seen, the diffraction fingerprint for the Rhombohedral and Hexagonal structures are very similar, hence they are clubbed together in one class. Therefore, the dataset essentially consists of structures represented as images with shape (64, 64, 3), alongwith their corresponding class labels (one of 7 classes, labels ranging from 0 to 6). To make it easier to deal with such class labels, one-hot encoding is used. Furthermore, an example of some defected structures alongwith their corresponding diffraction fingerprints have

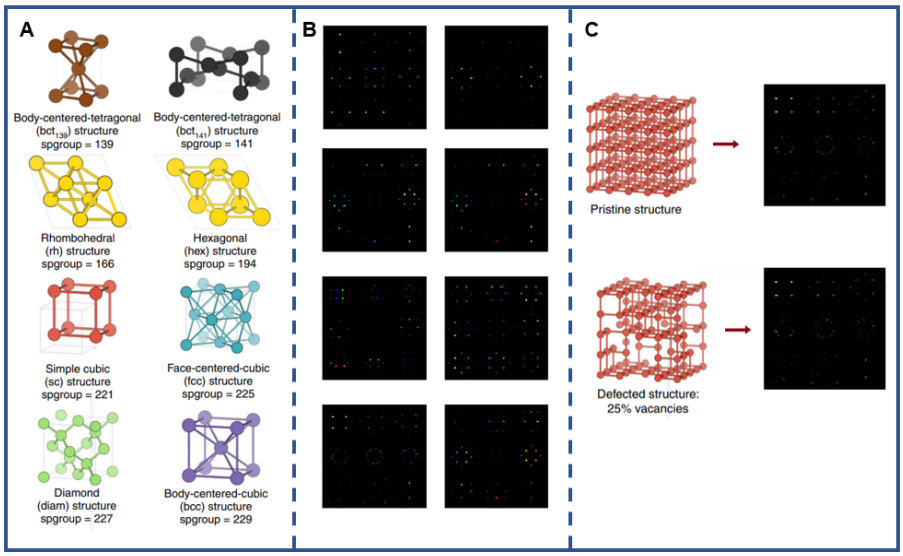

Figure 2: A: An example structure taken from each of the crystal systems considered. B: The corresponding 2D Diffraction fingerprint computed for each of the structures shown in A. C: A pristine cubic structure (without any defects) shown at the top; the same structure with 25% vacancies is shown at the bottom. The corresponding diffraction fingerprint is shown with the defected structures. Note: Zooming-in might be required to better observe the pattern.

been shown in Figure 2C. For this project, instead of generating data from scratch, the data generated in the reference paper [5] is directly used, also given in the GitHub repository[2].

2.3 Convolutional Neural Network

Since the dataset consists of images, a convolutional neural network (CNN) is used as a classifier. A schematic of the network is shown in Figure 3, and the model architecture used is given in Table 1. In such a network, the scalar product between the filter and the input at each place is computed after a learnable filter (also known as kernel) is convolved across the image. This generates a two-dimensional activation map for that filter at each spatial position, which is subsequently activated using a rectified linear unit (ReLu). To coarse grain the representation, a downsampling method (such as max pooling) is used. Six convolutional layers and two maximum pooling layers are placed sequentially. To finish the classification procedure, the output of the convolutional/downsampling layers sequence is passed to fully connected layers (which are regularised using dropout) and a final fully-connected layer with softmax ac-

[2]www.github.com/vir-k01/CH5650

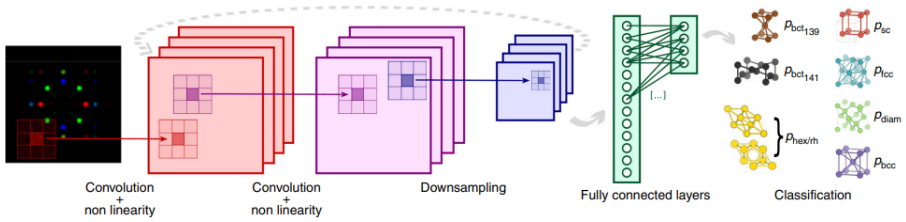

Figure 3: Schematic showing a convolutional neural network that takes the diffraction fingerprint as input and predicts the class label of the crystal structure the input diffraction fingerprint belongs to. (Image taken from [5])

Layer type	Specifications
Convolutional layer	Kernel: 7x7, 32 filters
Convolutional layer	Kernel: 7x7, 32 filters
Max pooling layer	Pool-size: 2x2, stride: 2x2
Convolutional layer	Kernel: 7x7, 16 filters
Convolutional layer	Kernel: 7x7, 16 filters
Max pooling layer	Pool-size: 2x2, stride: 2x2
Convolutional layer	Kernel: 7x7, 8 filters
Convolutional layer	Kernel: 7x7, 8 filters
Fully-connected + Dropout	Size: 128, dropout: 25%
Batch normalization	Size: 128
Fully-connected + softmax activation	Size: 7

Table 1: Description of the model architecture used for the CNN.

tivation. The CNN outputs the probabilities that the input image, and therefore the corresponding material, belongs to a given class. Minimizing the classification error, the above-mentioned filters are learned—through backpropagation and they will activate when a similar feature (e.g., edges or curves for initial layers, and more complex motifs for deeper layers) appears in the input. The training was performed using an Adam optimizer with batches of 32 images for 5 epochs. A learning rate of 1e-3 was used with categorical cross-entropy as the loss function. Training time was around 2.5 minutes on a Tesla K80 GPU provided in the Google Colab environment. The CNN was implemeted using the TensorFlow deep learning framework and the Keras API.

2.4 Dimensionality reduction and clustering

To extend the work, the obvious next step is: "Assuming that class labels do not exist for the data, is it still possible to classify it into different categories?" Such a problem can be possible when working with very large datasets of crystals, wherein annotating the samples could be very time consuming. Although solving problem this is the very purpose of the CNN model, training such a CNN model

Layer type	Nodes	Activation
Input Layer	12280	None
Encoding Layer 1	128	ReLu
Encoding Layer 2	64	ReLu
Bottleneck Layer	2	ReLu
Decoding Layer 1	64	ReLu
Decoding Layer 2	128	ReLu
Output Layer	12280	Tanh

Table 2: Description of the model architecture used for the simple autoencoder. All layers are fully-connected and the input image (which is of dimensions 64x64x3) is flattened before passing through the autoencoder. After training, only the first half (from the Input Layer to the Bottleneck Layer) is used as the encoder.

also requires atleast some labelled data (more than 10,000 samples in the above case), which can be a bottleneck in the ease and versatility of this approach. One way to eliminate the need for such labelled data is to use unsurpervised learning approaches such as clustering. Hence, the approach is as follows: (1) Reduce the dimensionality of the data (which is initially 64x64x3), down to a smaller number (say atmost 100); (2) Apply a clustering algorithm to classify the reduced representation of the data into 7 clusters (since there are 7 classes); (3) Classify new samples into one of these 7 clusters by comparing the Euclidean distances (which is essentially a generalization of the norm or cartesian distance in n dimensions) to the cluster centres in the reduced dimensional representation. The new sample is then assigned to the cluster centre it is "closest" to.

For step (1), several dimensionality reduction techniques have been employed, such as Linear PCA, Kernel PCA (with a polynomial kernel) and an autoencoder. For step (2), the K-Means clustering algorithm is used. Finally during step (3), several permutations of labelling each clustering is tried and the best fit is taken to be the classifcation made by this approach, and the accuracy of classification is then calculated. This has to be done since all we get from this approach is the existance of 7 clusters, and not what each cluster represents physically.

Both Linear and Kernel PCA (with polynomial kernel) reduce the dimensionality of the data down to 50 based on the scree plot of the PCA, which is shown in Figure 4. The architecture of a simple autoencoder used is given in Table 2. The autoencoder is trained for 10 epochs with 95% of the dataset used for training, using the Adam optimizer and mean squared error (MSE) as the loss function. The loss curves for the autoencoder are shown in Figure 5.

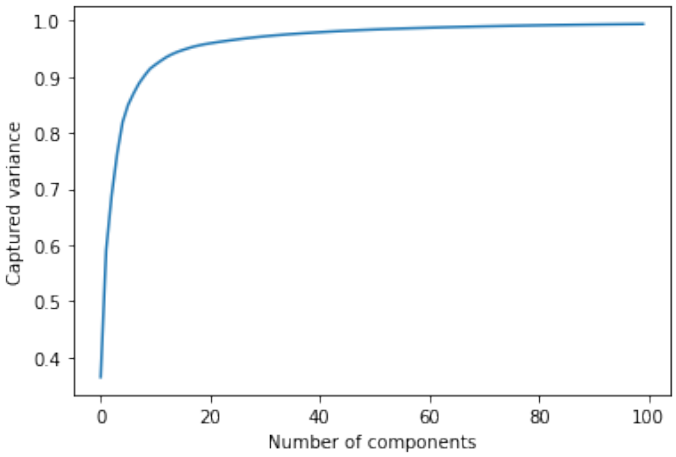

Figure 4: Scree plot obtained for the Linear PCA. As can be seen, only around 50 components need to be taken to capture $\geq$95% variance.

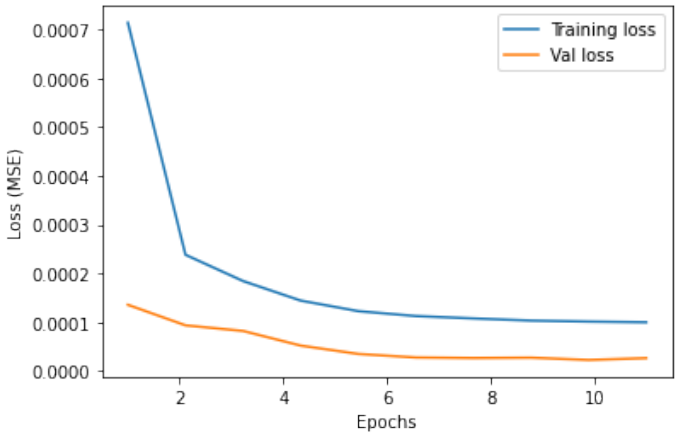

Figure 5: Loss vs epochs while training the autoencoder on the dataset. Mean Squared Error (MSE) is used as the loss function and 95% of the dataset was used for training and 5% for validation. As can be seen, the reconstruction loss at the end of 10 epochs is discernably low.

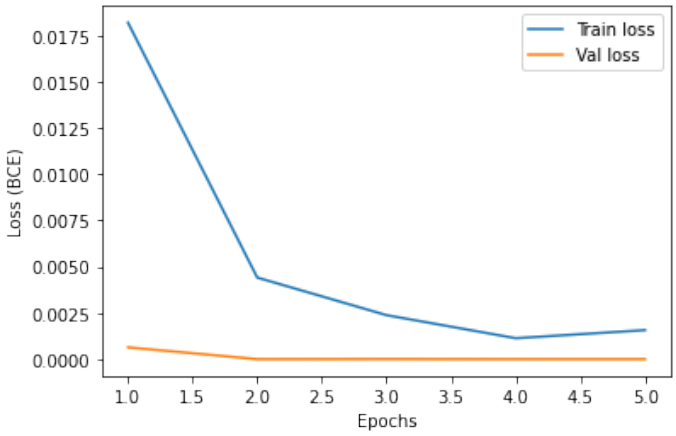

Figure 6: Loss vs epochs while training the CNN on the dataset. Cross-entropy loss was used as the loss function and 95% of the dataset was used for training and 5% for validation (same as the reference paper).

True(↓)/Predicted(→)	1	2	3	4	5	6	7
1	2601	0	0	0	0	0	0
2	0	221	0	0	0	0	0
3	0	0	842	0	0	0	0
4	0	0	0	910	0	0	0
5	0	0	0	0	2761	0	0
6	0	0	0	0	0	3178	0
7	0	0	0	0	0	0	4

Table 3: Confusion matrix for predictions made by the trained CNN model on the pristine (undamaged) dataset. Ideally, all entries should lie along the diagonal for perfect prediction, which does happen in this case.

3 Results and Discussion

3.1 Performance on dataset

The training curves (loss versus epoch) have been shown in Figure 6. The confusion matrix for the classification on the dataset is given in Table 3. As can be seen, since there are no off-diagonal entries, the model correctly classifies 100% of the samples. The same can be seen for the defected samples, as shown in Table 4, which is in fact identical to Table 3. Hence, similar to the reference paper, the trained model is also very accurate in its predictions on both pristine as well as defected crystals.

True(↓)/Predicted(→)	1	2	3	4	5	6	7
1	2601	0	0	0	0	0	0
2	0	221	0	0	0	0	0
3	0	0	842	0	0	0	0
4	0	0	0	910	0	0	0
5	0	0	0	0	2761	0	0
6	0	0	0	0	0	3178	0
7	0	0	0	0	0	0	4

Table 4: Confusion matrix for predictions made by the trained CNN model on the defected structures. Ideally, all entries should lie along the diagonal for perfect prediction, which does happen in this case.

3.2 Visualizing the network's predictions using attentive response maps

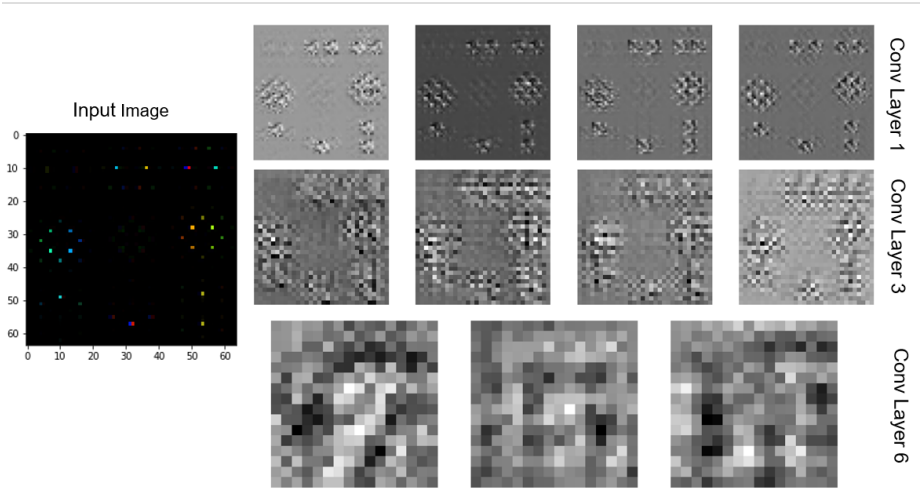

Figure 7: Visualization of the activations of the first 4 kernels of the 1st, 3rd and 6th Convolution layer in the trained CNN. These maps have been generated by passing a sample from the Rh/Hex class (Class 5) as input. The white regions have been "activated" by the kernel, hence, these can be said to be important in the classification done by the model.

As detailed in the reference papers[5, 1], the activations of the kernels in the CNN for a particular input can be visualised using the attentive response maps. Such activation maps can be obtained for each kernel of each convolution layer of the trained network by applying a linear activation after convolving the input of that layer with the trained kernel. For this report, the maps were

generated using the keract library [3]. As can be seen in Figure 7, which contains the attentive response maps corresponding to the first 4 kernels of the 1st, 3rd Conv layers and the first 3 kernels of the 6th Conv layer of the trained CNN when a sample from the Rh/Hex class (Class 5 of the dataset) is given as input. Similar to the reference paper[5], the hierarchical nature of the features learnt by the CNN can be observed. The earlier convolution layers learn to locate the positions of the various unit cells present in the diffraction pattern while the later layers learn to locate the diffraction peaks (white regions in the 6th Conv layer correspond approximately to the diffraction peaks of the input image).

3.3 Extension of model with unsupervised learning

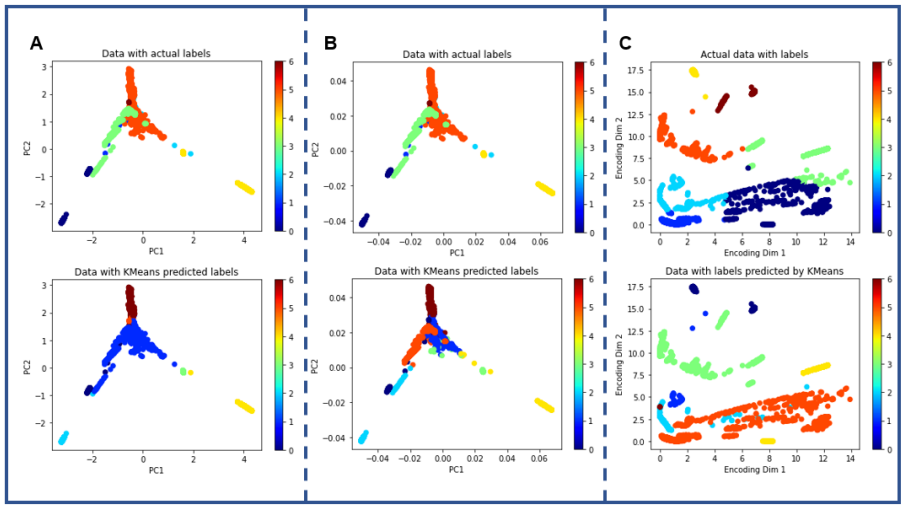

Figure 8: Reduced order representation of the dataset. Top row: Labelled using actual class labels, coloured accordingly. Bottom row: Labelled after clustering using K-Means, coloured accordingly. The reduced representations are- A: Linear PCA; B: Kernel PCA (Polynomial kernel); C: Autoencoder.

Model	Accuracy
Linear PCA + K-Means	84.09%
Kernel PCA + K-Means	84.07%
Autoencoder + K-Means	86.81%
CNN	**100%**

Table 5: Performance of the models used in this report.

The reduced dimensional representation (first 2 dimensions) is shown for each of the techniques used, alongwith the clusters predicted (color coded ap-

propriately) using K-Means in Figures 8A, B and C for Linear PCA, Kernel PCA and the Autoencoder respectively. Since we don't a-priori know what each cluster computed by K-Means represents physcially, the colour coding in the bottom row of figures of Figure 8 can be different from the top row. As of now, only the relative classification of the clusters has to be looked at, i.e, are the samples that are in the same cluster in the top row of Figure 8 (represented by the same colour) still present in the same cluster in the bottom row as well or not. The classification accuracy for each approach is given in Table 5. As can be seen, the best accuracy using the unsupervised learning approach is still far below that of the CNN. One reason for this could be the rather small size of the dataset, and the heavy class imbalance in the data (only 4 samples for BCC as compared to over 3000 samples for Rhombohedral). Another reason could be the fact that none of the lower dimensional representations have 7 clear distinct clusters visible, i.e., there is some overlap in the clusters, hence errors in classification are to be expected when using distance based metrics.

4 Conclusion

In this report, the CNN model proposed in the reference paper[5] was trained, using the 2D Diffraction fingerprints of 7 classes of crystals, and an accuracy of 100% was found when predicting for both pristine as well as defected crystals. Furthermore, the work was extended using unsupervised learning techniques to perform crystal structure classification given the 2D Diffraction fingerprints of the crystals. This was done to remove the need of having labelled data, which is a potential bottleneck in the CNN based approach. Hence, dimensionality reduction of the dataset was performed using PCA (Linear and Kernel) and a simple autoencoder and then clustering of the reduced order representations of the crystals was performed using the K-Means algorithm. This still only gave an accuracy of 86.81% (using the autoencoder and K-Means approach), which could be attributed to the heavy class imbalance of the dataset used and the overlapping of clusters of different classes in the reduced representation. Although this was only a short experiment to test out unsupervised learning on such a problem, it is clear that more data (for some classes of the dataset) and perhaps more complex, highly non-linear dimensionality reduction techniques (such as Convolutional autoencoders) could lead to better results.

References

[1] Devinder Kumar et al. *Understanding Anatomy Classification Through Attentive Response Maps*. 2016. DOI: 10.48550/ARXIV.1611.06284. URL: https://arxiv.org/abs/1611.06284.

[2] Andreas Leitherer, Angelo Ziletti, and Luca M. Ghiringhelli. "Robust recognition and exploratory analysis of crystal structures via Bayesian deep learning". In: *Nature Communications* 12.1 (Oct. 2021). DOI: 10.1038/s41467-021-26511-5. URL: https://doi.org/10.1038/s41467-021-26511-5.

[3] Philippe Remy. *Keract: A library for visualizing activations and gradients*. https://github.com/philipperemy/keract. 2020.

[4] Wahyu Setyawan and Stefano Curtarolo. "High-throughput electronic band structure calculations: Challenges and tools". In: *Computational materials science* 49.2 (2010), pp. 299–312.

[5] Angelo Ziletti et al. "Insightful classification of crystal structures using deep learning". In: *Nature communications* 9.1 (2018), pp. 1–10.