

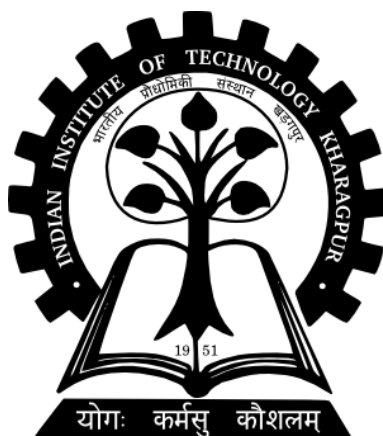
# **EMPLOYING MACHINE LEARNING TECHNIQUES TO PREDICT MATERIAL PROPERTIES FROM STRUCTURAL DATA**

Thesis submitted in complete fulfilment  
of  
**Masters in Technology (M. Tech.)**  
in  
**Chemical Engineering**

By

**SAURABH KUMAR PANDEY**  
**(17CH30055)**

*Under the guidance of*  
**PROF. SOMENATH GANGULY**



Department of Chemical Engineering  
Indian Institute of Technology, Kharagpur  
Kharagpur-721302

## DECLARATION

I certify that the work contained in this report entitled “**EMPLOYING MACHINE LEARNING TECHNIQUES TO PREDICT MATERIAL PROPERTIES FROM STRUCTURAL DATA**” is original and has been done by me under the guidance of my supervisor, Prof. Somenath Ganguly. The work has not been submitted for any purpose to any other university or institute. Whenever I have used materials (i.e. data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

Roll - 17CH30055

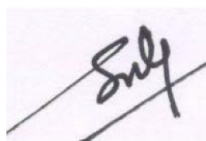
Saurabh Kumar Pandey

Department of Chemical Engineering,  
Indian Institute of Technology, Kharagpur

## CERTIFICATE

This is to certify that the thesis entitled “EMPLOYING MACHINE LEARNING TECHNIQUES TO PREDICT MATERIAL PROPERTIES FROM STRUCTURAL DATA”, has been submitted to the Indian Institute of Technology, Kharagpur, in partial fulfilment of Master of Technology in Chemical Engineering Department at the Indian Institute of Technology, Kharagpur.

It is a bonafide work carried out by Mr. Saurabh Kumar Pandey (Roll no. 17CH30055) under my supervision and guidance.

A handwritten signature in black ink, appearing to read 'Suly', is written over a light blue horizontal line.

PROF. SOMENATH GANGULY

DEPARTMENT OF CHEMICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

## ACKNOWLEDGEMENT

I am thankful to my supervisor **Prof. Somenath Ganguly** and my mentor **Mr. Smruti Ranjan Sethi** for their guidance throughout the research work. They have always helped me with their valuable suggestions, innovative ideas and technical expertise in shaping/molding and improving my knowledge and skills in the research area. I am highly obliged at the way he they have been patient and shown confidence in me as well as provided encouragement and mental support in carrying out the work during my difficult times. They have always been a source of inspiration. It has been an honor working/learning under their supervision.

I am thankful to my project guide for always guiding and supervising me, and correcting my mistakes every time. I would also like to thank my parents and friends who have always supported my work and me. Last but not the least, I am thankful to everyone, who directly or indirectly, helped me in completing this project.

Date: 29/04/2022

Saurabh Kumar Pandey

Place: Kharagpur, India

# CONTENTS

1. Abstract	6
2. Introduction	7
3. Effective Diffusivity of Porous Media	
3.1. Dataset	8
3.2. Data Pre-processing	9
3.3. Network Development	9
3.4. Hyperparameter Optimization	10
3.5. Training Plots	12
3.6. Results	16
4. Elastic Properties of Inorganic Solid Electrolytes	
4.1. Dataset	17
4.2. Crystal Graph Convolution Neural Networks	17
4.3. Crystal Graph Generation	18
4.4. CGCNN Network Architecture	20
4.5. Model Development	21
4.6. Hyperparameter Optimization	23
4.7. Results	23
5. Future Scope	
5.1. Porosity informed CNN	27
5.2. Pre-processed pore structures as input	27
5.3. Dendritic growth suppression in Li batteries	27
6. References	28
APPENDIX	29

## 1. ABSTRACT

The application of machine learning is getting into the roots of every domain and this work reports the application of machine learning methods to predict material properties from their structural data. This work specifically includes two important applications – predicting effective diffusivity of porous media and predicting elastic properties like Bulk modulus, Shear modulus from structured data of solid inorganic electrolytes. Convolutional Neural Networks based algorithms have gained wide importance for image-based methods and we have tried to leverage the power of CNNs to develop simple architectures for two different tasks.

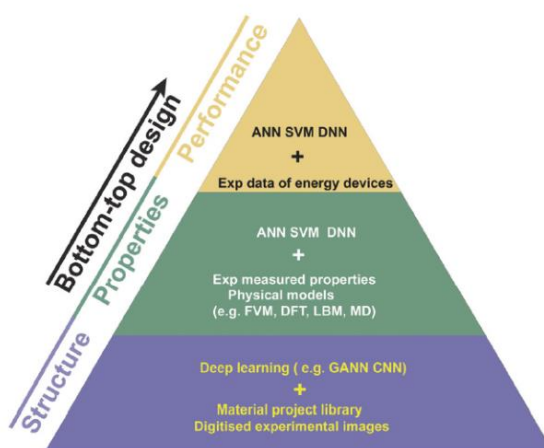
We develop a 2-layer neural network to predict effective diffusivity for porous media. The model built is optimized using hyperparameter tuning of different components of CNN like kernel size, number of filters, padding, activation functions, learning rate, etc. The optimized model has a relative error of ~30% on the validation set and ~39% on the test set. Further improvements in the score include more advanced methods such as introducing a physical factor like Porosity into the neural network and pre-processing the structure images to remove dead-end or trapped spaces to let the filters learn the features more suitably.

This work also extends the application of CNNs to learn from structural data of solid electrolytes. We present a generalized Crystal Graph CNN (CGCNN) where we directly build CNN on top of crystal graphs generated from crystal structures. Hyperparameter optimization leads us to quite good results where different parameters include learning rate, optimizers, number of convolutional layers, number of hidden layers, atom feature length vector, etc. We also show how this work can be applied to solve dendritic growth suppression which a major problem in lithium batteries.

*Keywords:* Porous media, Solid Inorganic Electrolytes, Effective Diffusivity, Bulk modulus, Shear modulus, Convolutional Neural Network, Crystal Graph CNN, Machine Learning, Hyperparameter tuning

## 2. INTRODUCTION

This work underlines the application of machine learning methods for predicting the effective diffusivity ( $De$ ) of two-dimensional porous media from images of their structures<sup>[1]</sup> and use of graph based CNNs to predict elastic properties from the structure data. The datasets generated from (lattice Boltzmann) LBM simulations and those available on Materials API are used to train convolutional neural network (CNN) models and evaluate their performance.



**Fig.1 - AI-enabled structure reconstruction and generation, prediction of properties and in-service performance for porous energy materials**

The effective transport properties of porous media are often computed using empirical correlations or effective medium theories with their structure information<sup>[2]</sup> (e.g., porosity) as input. Such an approach needs little computational cost and can be very accurate for some specific (often idealized) classes of porous media. It remains a great challenge to develop methods for predicting the effective transport properties of porous media that require low computational cost but offer high accuracy for diverse porous structures.

Machine learning can potentially be an effective approach for tackling the above challenge. Deep neural networks have demonstrated good predictive power when their input and output have important correlation with each other. Furthermore, image-based learning has been shown to be able to extract important physical features from images<sup>[3]</sup>. Because the effective transport properties (in particular, the effective diffusivity) of porous media is largely determined by their structure which can be conveniently represented using their binary images, conceivably, one can develop a surrogate deep learning model to extract key geometrical features from images of porous media and predict their transport properties. The prime objective of the work is stated as follows:

- Study of machine learning based approach to study different properties of porous media from structural images.
- Optimization of hyperparameters of a convolutional neural network to predict effective diffusivity of porous media.
- Employing Graph based methods to predict elastic properties like Bulk and Shear modulus for materials from their structural data.

### 3. Effective Diffusivity of Porous Media

#### 3.1. Dataset

Generated dataset is in CSV format as shown below. The file contains 301 binary images each with 16384 pixels (128\*128) with pixel values of 0 and 1

	0	1	2	3	4	5	6	7	8	9	...	291	292	293	294	295	296	297	298	299	300
0	0	1	1	0	1	0	1	1	1	0	...	1	1	1	0	0	1	0	1	0	0
1	0	1	1	0	1	0	1	1	1	0	...	1	1	1	0	0	1	0	1	0	0
2	0	1	1	0	1	0	1	1	1	0	...	1	0	1	0	0	1	0	1	0	0
3	0	1	1	0	1	0	1	1	1	0	...	1	0	1	0	0	0	0	1	0	0
4	0	1	1	0	1	0	0	0	1	0	...	1	0	1	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16379	0	1	0	0	0	0	0	0	1	1	...	0	0	0	1	1	0	0	0	1	0
16380	0	1	0	0	0	0	0	0	1	1	...	0	0	0	1	1	0	0	0	1	0
16381	0	1	0	0	0	0	0	0	1	1	...	0	0	0	1	1	0	0	0	1	0
16382	0	1	0	0	0	0	0	0	1	1	...	0	0	0	1	1	0	0	0	1	0
16383	0	1	0	0	0	0	0	0	1	1	...	0	0	0	1	1	0	0	0	1	0

16384 rows × 301 columns

The above dataset is converted into 128\*128 binary images using OpenCV library in Python. Some sample images are shown below.



Fig. 2 - Sample dataset images. White color denotes pore space and black color denotes solid phase

The diffusivity values in the dataset can be described as follows. These values are normalized by a factor ( $1.958e-12$ ) to get effective diffusivity for the porous media.

Mean	$7.607423e-14$
Std	$3.261657e-14$
Min	$7.837232e-15$
Max	$2.948378e-13$

Table. 1 – Description of Effective Diffusivity data



### 3.2. Data Pre-processing

**Normalization** – Since the diffusivity values are small, they are normalized by a factor ( $1.958 \times 10^{-12}$ ) to get all the values in the range 0-1

**Train/Val/Test split** – The data is split into train (80%), validation (10%) and test (10%). Training data is used to train images, validation data is used to select the model with best set of hyperparameters and test set is used to evaluate the model on test data.

### 3.3. Convolutional Neural Network Development

Of the many deep neural network models, convolutional neural network (CNN) is commonly applied to analyse visual imagery and has achieved much success in image classification. Recently, CNN has also been adopted to study the effective properties of complex materials and showed much potential for efficient and accurate prediction of a material's effective properties from its structure. Prediction of permeability from images of porous media using CNN has provided useful insights in understanding the correlation between geometric features and transport properties.

A simple 2-layer CNN model is developed to study the images and derive physical insights from structures and these insights are used to develop a model that can be used to predict effective diffusivity of any given unknown porous media.

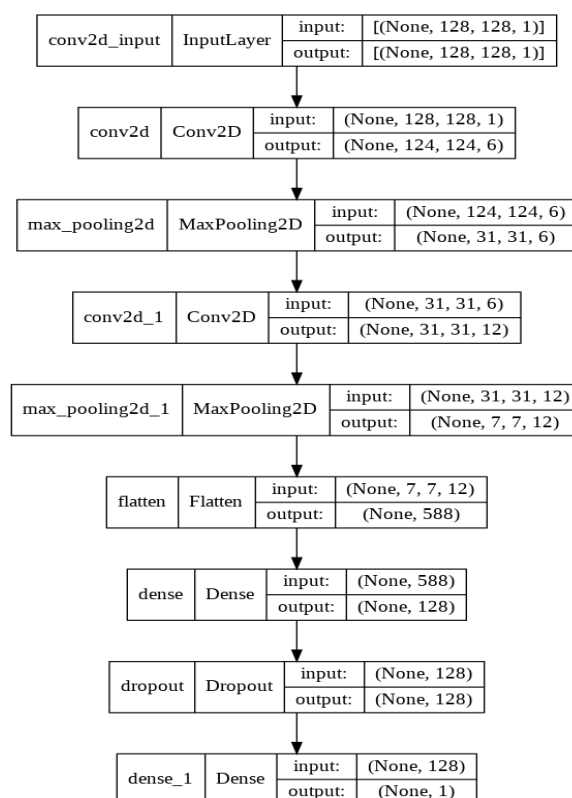


Fig. 3 - A simple 2-layer CNN model architecture.

Dense layer represents the Fully Connected layers and dense\_1 is the final output to regress the model

## Some CNN Terminologies

1. **Sequential model** – this model is used for a plain stack of layers where each layer has exactly one input tensor and one output tensor
2. **Convolution layers** - This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.
3. **Max Pooling layers** - Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window for each channel of the input.
4. **ReLU Activation function** – Applies rectified linear unit activation :  $\max(x, 0)$  on the input tensor.
5. **Padding** - one of "valid" or "same" . "valid" means no padding. "same" results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.
6. **Filters** - the dimensionality of the output space (i.e. the number of output filters in the convolution).
7. **Kernel size** - An integer specifying the height and width of the 2D convolution window.
8. **Dropout** - The Dropout layer randomly sets input units to 0 during training time, which helps prevent overfitting.
9. **Dense layers** - Dense layer is the Fully connected layer which implements the dot and multiplication operations on weight, input and bias matrices.
10. **Loss function** - Loss function used to regress the CNN model is Mean Squared Error (MSE), where  $y$  and  $y'$  are target and predicted values respectively.

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

11. **Performance Metric** – Relative error is used to measure and compare performance of different models.

$$RE = \frac{\text{Predicted value} - \text{Target value}}{\text{Target value}}$$

### 3.4. Hyperparameter Optimization

<b>Kernel size</b>	(2,2)	(3,3)	(4,4)	(5,5)
<b>Learning Rate</b>	1e-2	1e-3	1e-4	
<b>Number of filters</b>	(8,16)	(16,32)	(32,64)	
<b>Strides</b>	(2,2)	(3,3)	(4,4)	
<b>Pool size</b>	(2,2)	(3,3)	(4,4)	
<b>Padding</b>	“same”	“valid”		
<b>Activation</b>	ReLU	Linear		

Table 2 – Hyperparameters to be tested with CNN model

a. Linear Activation function

<b>Kernel size</b>	(4,4)
<b>Learning Rate</b>	1e-2
<b>Number of filters</b>	(8,16)
<b>Strides</b>	(3,3)
<b>Pool size</b>	(3,3)
<b>Padding</b>	“valid”
<b>Activation</b>	Linear

**Table 3 – Hyperparameter configuration for best model with Linear Activation function**

<b>Train Set</b>	35.79
<b>Validation Set</b>	30.73
<b>Test Set</b>	39.71

**Table 4 – Train, Val and Test Relative Error for CNN with Linear Activation function**

b. ReLU activation function

<b>Kernel size</b>	(5,5)
<b>Learning Rate</b>	1e-2
<b>Number of filters</b>	(32,64)
<b>Strides</b>	(3,3)
<b>Pool size</b>	(2,2)
<b>Padding</b>	“valid”
<b>Activation</b>	ReLU

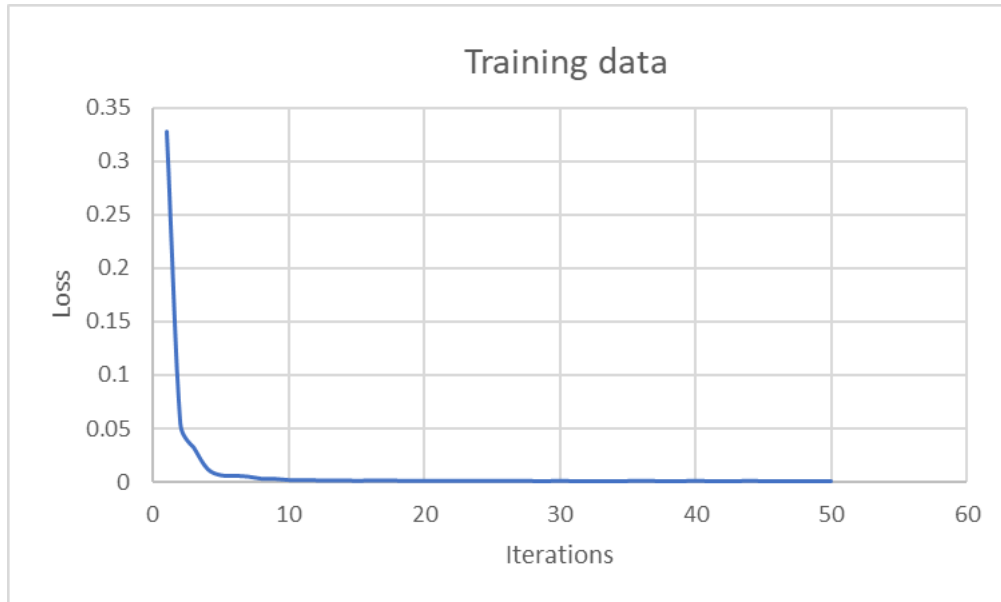
**Table 5 – Hyperparameter configuration for best model with ReLU Activation function**

<b>Train Set</b>	39.66
<b>Validation Set</b>	36.39
<b>Test Set</b>	45.55

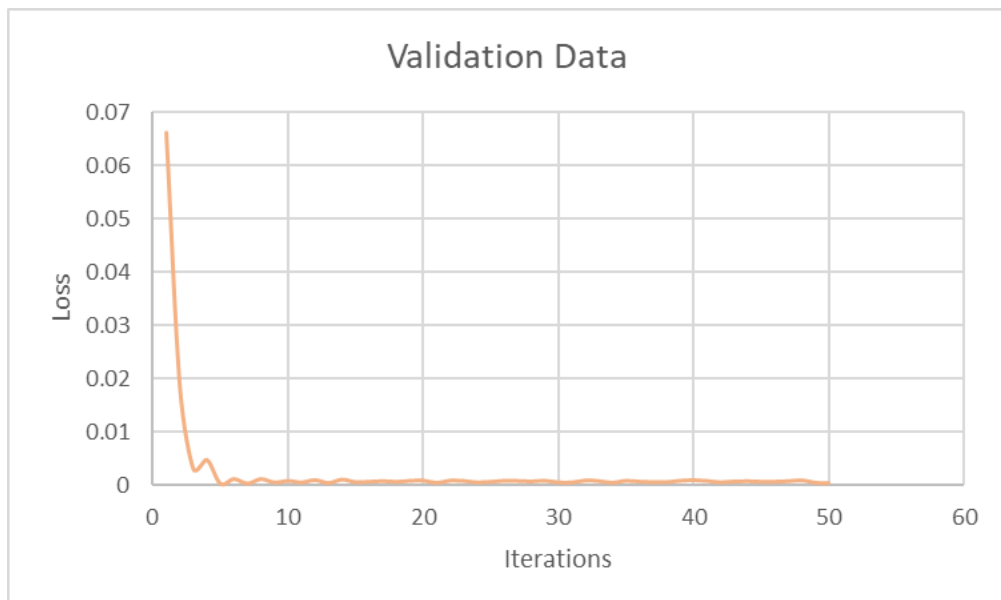
**Table 6 – Train, Val and Test Relative Error for CNN with ReLU Activation function**

### 3.5. Training Plots

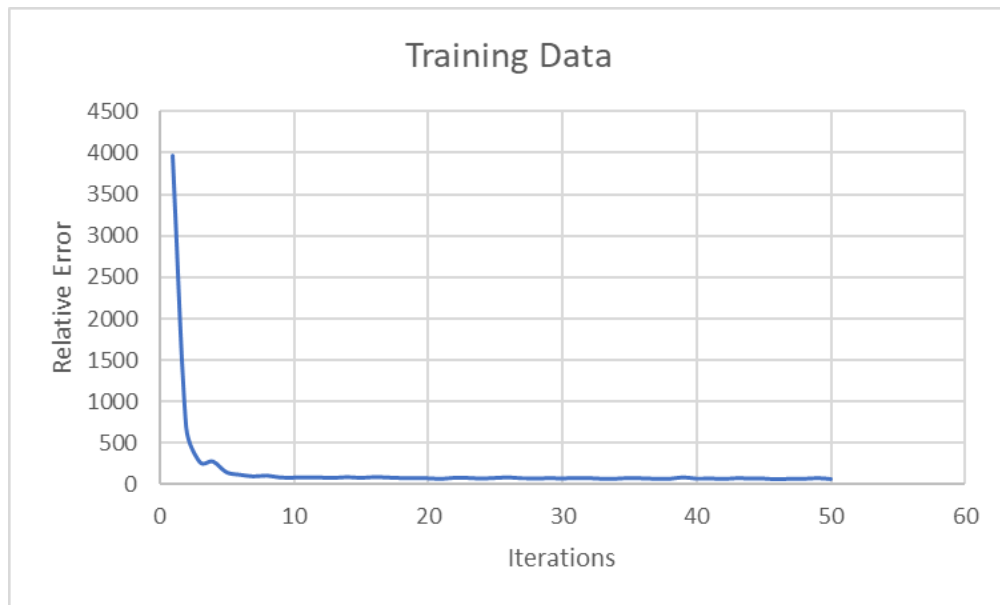
#### a. Linear Activation Function



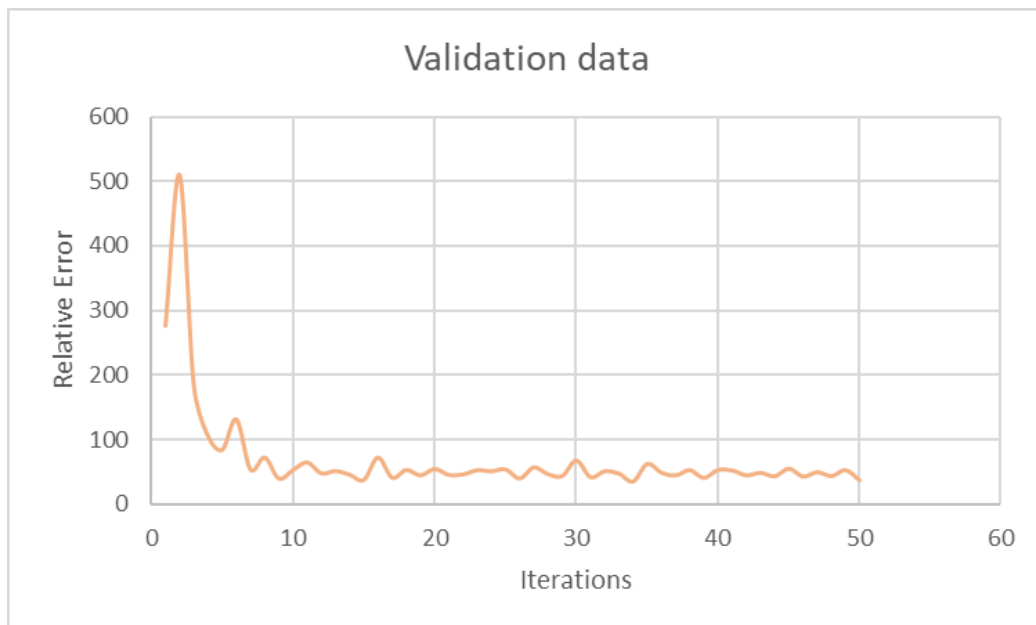
**Fig. 4 – Loss v/s Iterations for training data**



**Fig. 5 – Loss v/s Iterations for validation data**



**Fig. 6 - Relative error v/s Iterations for training data**

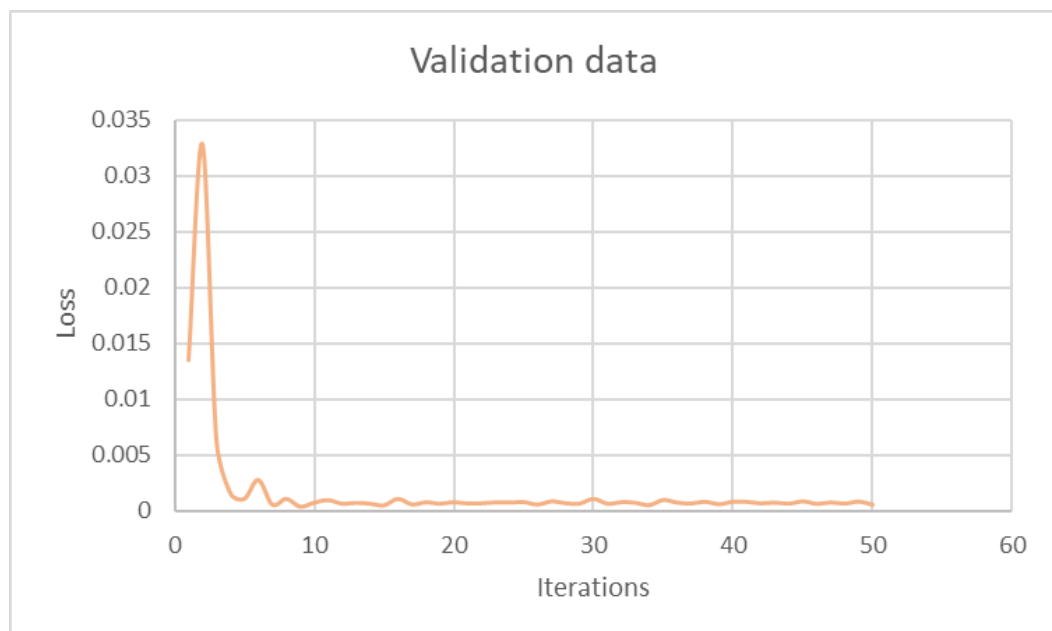


**Fig. 7 – Relative error v/s Iterations for validation data**

b. ReLU Activation function



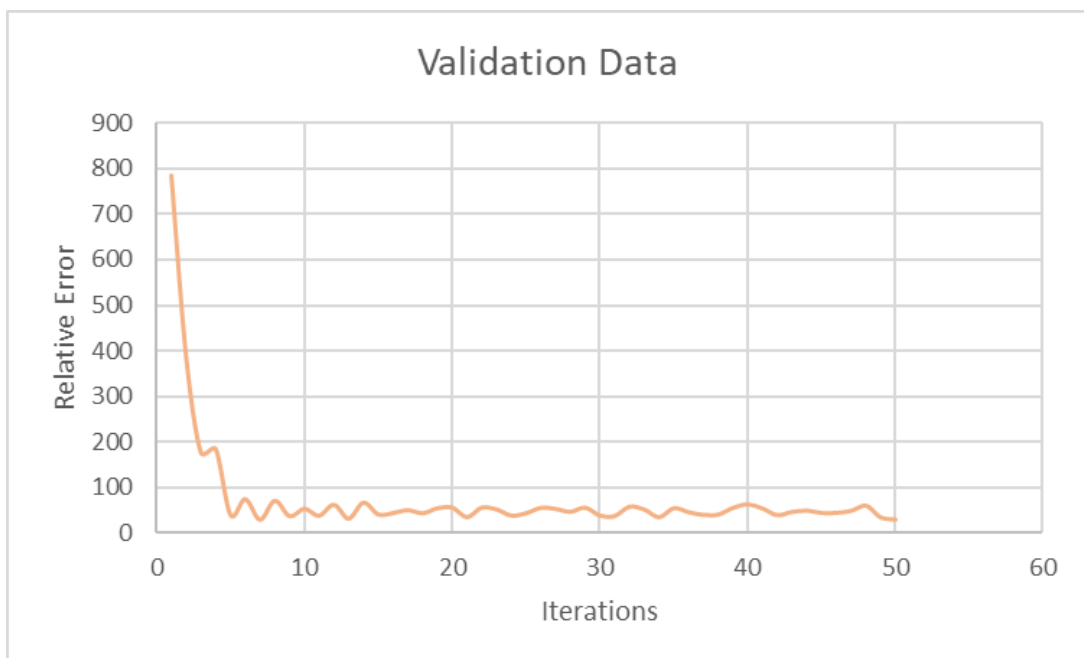
**Fig. 8 – Loss v/s Iterations for training data**



**Fig. 9 – Loss v/s Iterations for validation data**



**Fig. 10 – Relative error v/s Iterations for training data**



**Fig. 11 – Relative error v/s Iterations for validation data**

### 3.6. Results

In this section, we highlight all the details of the dataset, the architecture used, hyperparameter optimization and the final resultant model that we have to predict effective diffusivity of porous media.

Dataset contains binary images in the form of csv file as input and diffusivity values as targets. The targets are normalized to transform diffusivity to effective diffusivity and the data is then split into train, validation and test data with each being 90%, 10% and 10% respectively. Since the dataset contains only 300 images with rather simple features, we do not go for pretrained SOTA architectures like ResNet, VGG, EfficientNet, etc. and we rather stick to a basic 2-layer convolutional neural network to train the model.

We treat all the functionalities of this network like kernel size, number of filters, padding, activation function, pool size, learning rate, etc. as hyperparameters and we tune this set of hyperparameters to get the best neural network for the given data. As a part of hyperparameter tuning, we train close to 900 different 2-layer networks varying the parameters of the CNN and we finalize 2 models, one with a Linear Activation function and other one with a ReLU activation function. We get superior results with Linear Activation function but with more complex features, ReLU may be quite useful and hence we keep two models to predict effective diffusivity. We also observe that a larger filter (4x4, 5x5) provides us with better results as compared to a smaller filter (3x3), likely because the narrow filters cannot capture some important features spanning moderate to large numbers of pixels. Similarly, all other hyperparameters are chosen and varied.

With Linear Activation function, we get a relative error of 30.73% on validation dataset and a relative error of 39.71% on test dataset as specified in Table 4. With ReLU Activation function, we get a relative error of 36.39% on validation dataset and a relative error of 45.55% on test dataset as specified in Table 6. With these results in hand, we have made significant improvement in Relative Error and we further try to improve the results by using more suited neural network for the given data that may be able to capture the features more significantly and produce better results. We will further try to train a neural network with porosity values that will ingest a physical viewpoint to the whole neural network and may be helpful in further improving the results.



## 4. Elastic Properties of Inorganic Solid Electrolytes

### 4.1. Dataset

We use Materials API<sup>[11]</sup> to download the dataset. The database has around 46744 materials covering 87 elements, 7 lattice systems and 216 space groups. We observe that the elastic properties, bulk and shear moduli to be specific, has not been calculated for all these 46744 materials. The elastic tensor has only been calculated for around 9507 materials and for the rest of the materials, we have some statistical based models from which we get approximate values. Since the paper<sup>[6]</sup> states that they use only 2041 crystals for the modelling, we randomly take ~25% of 9507 materials to build our CGCNN for the bulk and shear modulus.

The construction of dataset basically consists of three simple steps

- (i) Download the conventional CIF files from the materials API which contains the structure of the materials
- (ii) Get the properties (bulk and shear modulus) from the materials API for the materials whose elastic tensor has been calculated
- (iii) Initialize a vector of features for all the atoms

Once we have these files with us, we can use the generalized CGCNN<sup>[6]</sup> to train the models for predicting bulk and shear modulus separately.

### 4.2. Graph Convolution Networks

GCN<sup>[5]</sup> are a very powerful neural network architecture for machine learning on graphs. In fact, they are so powerful that even a randomly initiated 2-layer GCN can produce useful feature representations of nodes in networks. The figure below illustrates a 2-dimensional representation of each node in a network produced by such a GCN. Notice that the relative nearness of nodes in the network is preserved in the 2-dimensional representation even without any training.

GCNs perform similar operations as convolutions in CNN where the model learns the features by inspecting neighboring nodes. The major difference between CNNs and GCNs is that CNNs are specially built to operate on regular (Euclidean) structured data, while GCNs are the generalized version of CNNs where the numbers of nodes connections vary and the nodes are unordered (irregular on non-Euclidean structured data).

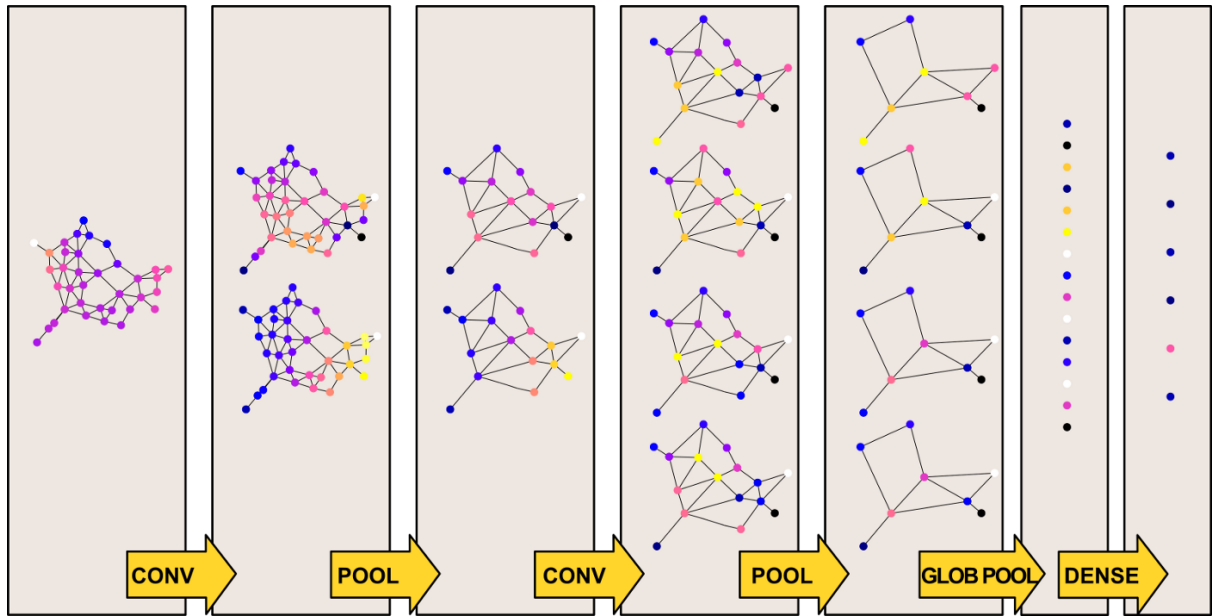


Fig. 12 - Graph Convolution Network

In the case of a GCN, our input is represented by the following elements:

- $N * C$  array  $x$  containing  $C$  features for each of the  $N$  nodes of the graphs
- $N * N$  adjacency matrix

We extend this knowledge and intuition of Graph Convolution Network to crystals which are called Crystal Graph Convolution Neural Network.

### 4.3. Crystal Graph Generation

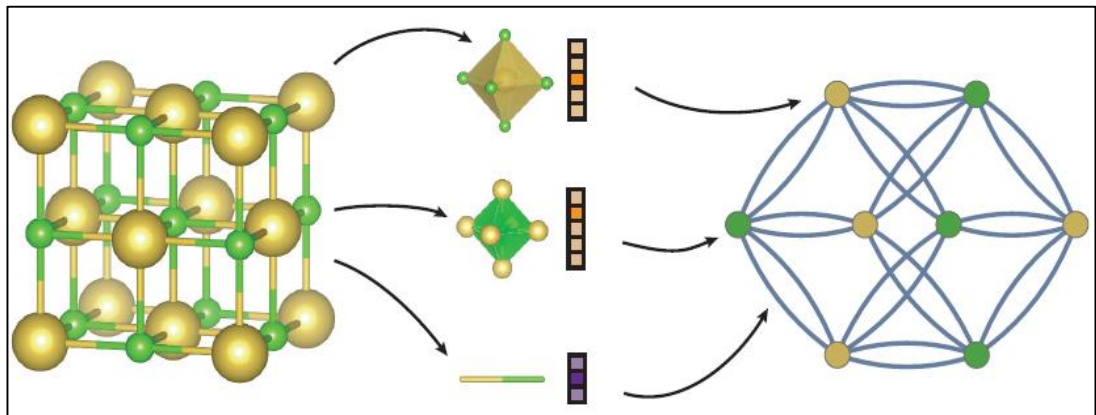


Fig. 13 - Construction of the crystal graph from structural data

Generation of crystal graphs from structural data is a challenging task before we feed this graph into Graph Convolution Networks. To accomplish this, we require a feature vector for all the atoms and the feature vector of the bond connecting atoms  $i$  and  $j$ . As illustrated in Fig. 13, a crystal graph  $G$  is an undirected multigraph which is defined by nodes representing atoms and edges representing connections between atoms in a crystal. The crystal graph is unlike normal graphs since it allows multiple edges between the same pair of end nodes, a characteristic for crystal graphs due to their periodicity, in contrast to molecular graphs. Each node  $i$  is represented by a feature vector  $v_i$ , encoding the property of the atom corresponding to node  $i$ . Similarly, each edge  $(i,j)_k$  is represented by a feature vector  $u(i,j)_k$  corresponding to the  $k$ -th bond connecting atom  $i$  and atom  $j$ .

The structural data for materials can be found in different forms. Crystallographic Information files (cif) have proved to be a big boost as far as the ease of crystal graph generation is considered but several materials and compounds do not have their structural data available in cif format at Material API database. One of the other most prominent notations is SMILES (Simplified Molecular Input Line Entry System) where we get the material notations in the form of strings. Since a CNN expects an unstructured data, we need to transform this notation to a sequence of feature vectors representing the symbols that occur in SMILES string<sup>[14]</sup>.

First, the input compound is represented by a SMILES string. Next, for each symbol in the SMILES string, a feature vector that is a distributed representation of the symbol is calculated. Each feature vector consists of 42 features, of which 21 features are used as symbols for atoms, and the remaining 21 features are used for original SMILES symbols. Each dimension in the 21-dimensional vector for an atom consists of the type of atom, and its degree, charge, and chirality, and each 21-dimensional vector for an original SMILES symbol is a one-hot vector that is a distributed representation of 21 original SMILES symbols. Note that one-hot vector is a binary vector with a single high (1) bit and all the others low (0). The length of the feature matrix is set to the maximum length of SMILES strings for compounds in a given dataset. In the feature matrix for SMILES strings of which the length is shorter than the maximum length, all the blank parts were padded with 0 to retain the input size. There may be other forms of representation as well which may require different kind of analysis and representation of features vectors before feeding them to CNN. In any kind of representation, we need to study the atoms being involved, their properties and how different atoms are connected to each other so that it is easy to decide the algorithm to build the associated crystal graphs to it.

#### 4.4. CGCNN Network Architecture

We present a generalized crystal graph convolutional neural networks (CGCNN) framework for representing periodic crystal systems that provides both material property prediction with DFT accuracy and atomic level chemical insights. The main idea in our approach is to represent the crystal structure by a crystal graph that encodes both atomic information and bonding interactions between atoms, and then build a convolutional neural network on top of the graph to automatically extract representations that are optimum for predicting target properties by training with DFT calculated data.

The convolutional neural networks built on top of the crystal graph consists of two major components: convolutional layers and pooling layers. Similar architectures have been used for computer vision, natural language processing, molecular fingerprinting, and general graph-structured data. The convolutional layers iteratively update the atom feature vector  $v_i$  by convolution with surrounding atoms and bonds with a non-linear graph convolution function.

$$v_i^{(t+1)} = \text{Conv} \left( v_i^{(t)}, v_j^{(t)}, u_{(i,j)_k} \right), (i,j)_k \in \mathcal{G}$$

After  $R$  convolutions, the network automatically learns the feature vector  $v_i^{(R)}$  for each atom by iteratively including its surrounding environment. The pooling layer is then used for producing an overall feature vector  $v_c$  for the crystal, which can be represented by a pooling function

$$v_c = \text{Pool} \left( v_0^{(0)}, v_1^{(0)}, \dots, v_N^{(0)}, \dots, v_N^{(R)} \right)$$

In addition to the convolutional and pooling layers, two fully-connected hidden layers with the depth of  $L_1$  and  $L_2$  are added to capture the complex mapping between crystal structure and property. Finally, an output layer is used to connect the  $L_2$  hidden layer to predict the target property.

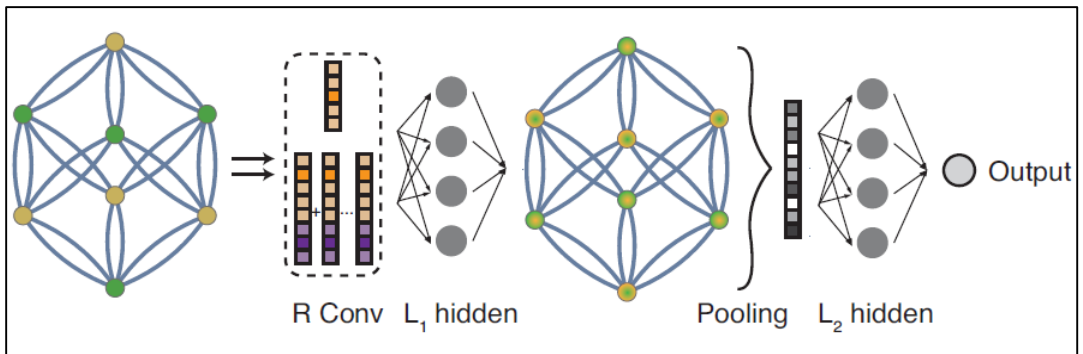


Fig. 14 - Convolutional neural network on top of the crystal graph.

The training is performed by minimizing the difference between the predicted property and the DFT calculated property  $y$ , defined by a cost function  $J(y, y')$ . The whole CGCNN can be considered as a function parameterized by weights  $W$  that maps a crystal  $C$  to the target property. Using backpropagation and stochastic gradient descent (SGD), we can solve the following optimization problem by iteratively updating the weights with DFT calculated data.

In the context of computational materials science, first principles DFT calculations allow the prediction and calculation of material behavior on the basis of quantum mechanical considerations, without requiring higher-order parameters such as fundamental material properties. In contemporary DFT techniques the electronic structure is evaluated using a potential acting on the system's electrons. This DFT potential is constructed as the sum of external potentials  $V_{\text{ext}}$ , which is determined solely by the structure and the elemental composition of the system, and an effective potential  $V_{\text{eff}}$ , which represents interelectronic interactions. Thus, a problem for a representative supercell of a material with  $n$  electrons can be studied as a set of  $n$  one-electron Schrödinger-like equations<sup>[15]</sup>.

## 4.5. Model Development

The code for the CGCNN model discussed has been written in PyTorch and the source code is available [here](#)<sup>[12]</sup>. The main file loads the structure files in CIF format. The CIFData dataset is a wrapper for a dataset where the crystal structures are stored in the form of CIF files. The dataset has the following directory structure.

```
root_dir
|---- id_prop.csv (csv containing the material IDs and properties)
|---- atom_init.json (json with feature vectors of different atoms)
|---- id0.cif
|---- id1.cif
.....
```

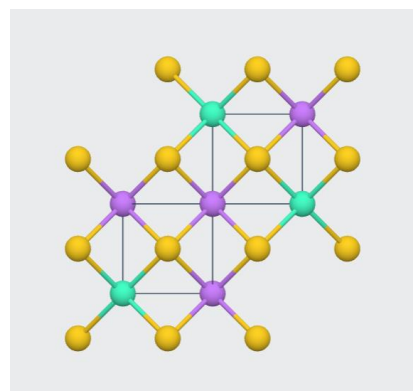
The PyTorch dataloader then loads the data files and splits the data into train, validation and test in the ratio of 60%, 20% and 20% respectively. In our case, we trained our model on ~2376 materials and performed validation and testing on ~760 data points. We then created a Crystal Graph Convolutional Neural Network which takes atom feature length, neighbour feature length, number of hidden layers before and after pooling and other intuitive parameters to create our CGCNN. The model summary for default parameters with four convolutional layers can be seen in APPENDIX.

The CGCNN model is then trained using MSE loss function and Adam/SGD optimizer for around 50 epochs and the best checkpoint is saved for further testing on unseen data.

We show the workflow of the whole CGCNN model by taking an example of a material from the database. We take LiGdAu<sub>2</sub> (mp-867244) as an example.

#### Lattice Parameters

$$\begin{aligned} a &= 6.1916 \text{ \AA} \\ b &= 6.916 \text{ \AA} \\ c &= 6.916 \text{ \AA} \\ \alpha &= 90.00^\circ \\ \beta &= 90.00^\circ \\ \gamma &= 90.00^\circ \end{aligned}$$



LiGdAu<sub>2</sub> structure

The Crystallographic Information File (CIF) contains the complete data of the material such as the lattice parameters stated above, the lattice volume, chemical formula and the atom coordinates are stored using loops. The CIFDataset class takes a CIF file (CIF file of LiGdAu<sub>2</sub> is in APPENDIX) as an input and builds a crystal graph using some maximum number of neighbours (12) and the cut-off radius for searching neighbours (8 Å). It returns a tuple of arrays with shapes ((16,92), (16,12,41), 16,12)) which represents atom feature vector, neighbour feature vector and neighbour index along with target and cif\_id of the material. The code uses Gaussian Distance to decide the neighbours.

Once we have this crystal graph generated, we use a utility function to load and divide the dataset into train, validation and test datasets using PyTorch Dataloader library. We divide the dataset into 60,20,20 split and use a collate\_pool function to divide the data into batches of 256 while training.

We then initialize a Crystal Graph Convolution Network using the atom feature vectors, neighbour feature vectors, number of convolution layers, number of hidden layers, etc. The shapes of LiGdAu<sub>2</sub> after each pass through the layers is tabulated in Table 7. The table shows the dimensions for a batch size of 256 with 16 atoms in the crystal.

layer	input_shape	output_shape
<i>Linear</i>	(256,16,92)	(256,16,128)
<i>Conv_1</i>	(256,16,128)	(256,16,64)
<i>Conv_2</i>	(256,16,64)	(256,16,32)
<i>Conv_3</i>	(256,16,32)	(256,16,16)
<i>Pooling</i>	(256,16,16)	(256,16,16)
<i>FC</i>	(256,16,16)	(256,16,32)
<i>Output</i>	(256,16,32)	(256,16,1)

Table 7 – Different layers and their input and output shapes in a CGCNN

## 4.6. Hyperparamter Optimization

From the above description of CGCNN, we can see that there are a lot of hyperparameters involved, both in the network architecture as well as model training. On the architecture front, the number of convolutional layers, the number of hidden layers before and after pooling, number of neurons in the atom features, the activation functions, etc are the parameters which we need to have a close look at and decide the best suitable combination of all. From the perspective of training, we need to decide the suitable learning rate, the optimizer being considered and the loss function.

We use Mean Absolute Error (MAE) as our performance metric to minimize the mean squared error loss function. We tried out different hyperparamters and below is a subset of those trials with the MAE on the test set.

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

The best performing model with an MAE(b) of 0.093 for bulk\_modulus and an MAE(s) of 0.149 for shear\_modulus has the following hyperparameters:

1. Number of convolutional layers (n\_conv) - 3
2. Number of hidden layers before pooling (n\_h1) - 1
3. Number of hidden layers after pooling (n\_h2) - 1
4. Length of learned atom feature vector (a\_fea\_len) - 128
5. Learning rate (LR) - 0.01
6. Optimizer (Optim) - Adam

A subset of different trials performed for hyperparameter tuning using CGCNN to predict bulk modulus and shear modulus has been appended as Table 10 in APPENDIX section of the report.

## 4.7. Results

Since a machine learning methodology employs a huge amount of assumptions and simplifications, we need to perform rigorous testing of the models that we have either on some unseen data present online or using some synthetic techniques to generate the data and test it. Machine Learning models demands generalizability and that can only be verified by performing testing on unseen real data. Fortunately we have a lot of data available since we used only 25% of the data for training. We report the results in terms of RMSE to compare our results with the ones reported by Ahmad et al<sup>[7]</sup>.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

1. We used the pretrained model to predict on the test data of ~760 materials available from materials API

	<b>RMSE [log(K)]</b>	<b>RMSE [log(G)]</b>
Ahmad et al.	0.1013	0.1268
Our work	0.1437	0.1680

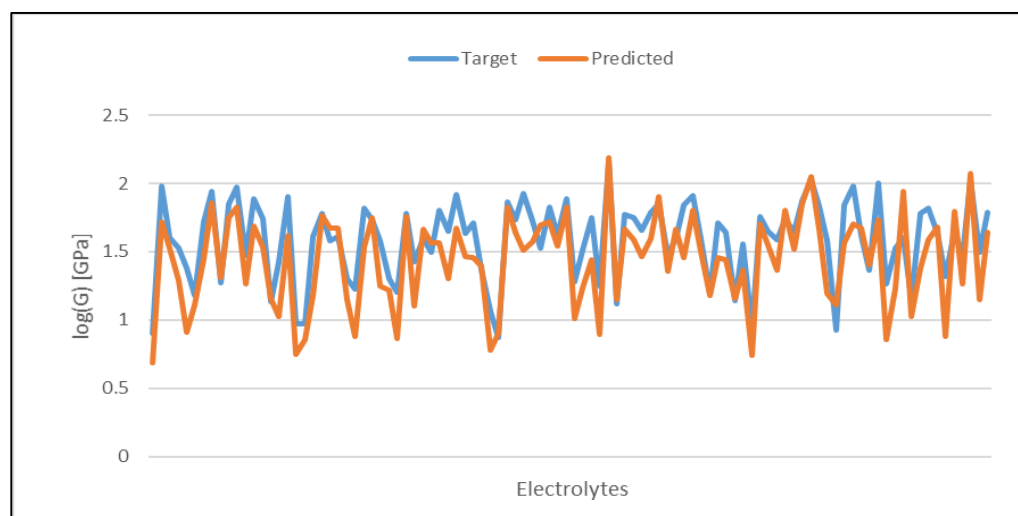
**Table 8 – Comparison of RMSE in log(GPa) for Shear and Bulk moduli**

We observe that our work produces RMSE values very close to the ones reported by Ahmad et al. The difference in the values can be attributed to the difference in the materials being used for training in the two works since the dataset is not mentioned anywhere explicitly.

2. Since a large amount of data had elastic tensors calculated using statistical models, we use around 100 materials' data to see how close our model is with respect to predictions from statistical models.

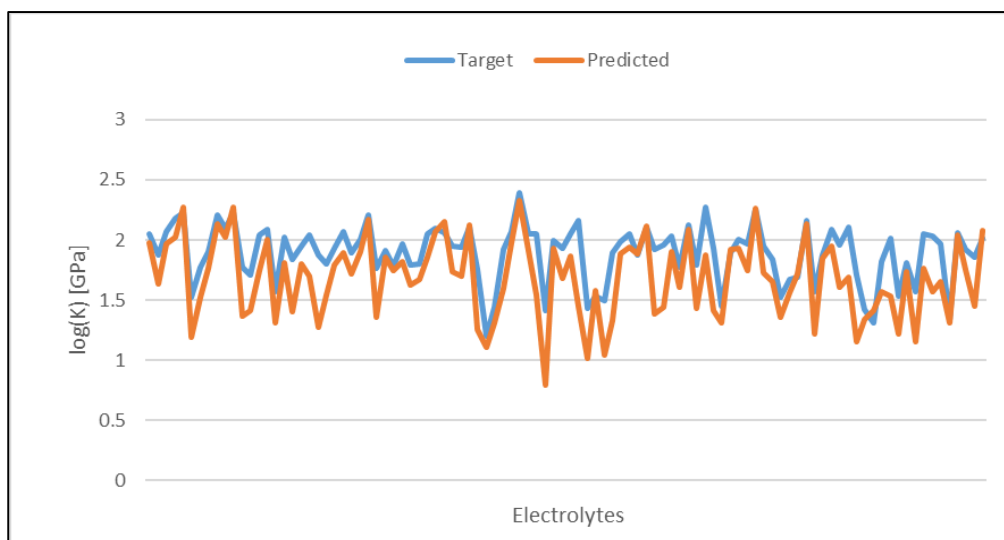
	<b>RMSE [log(K)]</b>	<b>RMSE [log(G)]</b>
Ahmad et al.	0.1013	0.1268
Our work	0.1437	0.1680
Statistical data	0.2773	0.2187

**Table 9 – Comparison of RMSE in log(GPa) for Shear and Bulk moduli**



**Fig. 15 - Graph showing overlap of target and predicted values of bulk modulus**



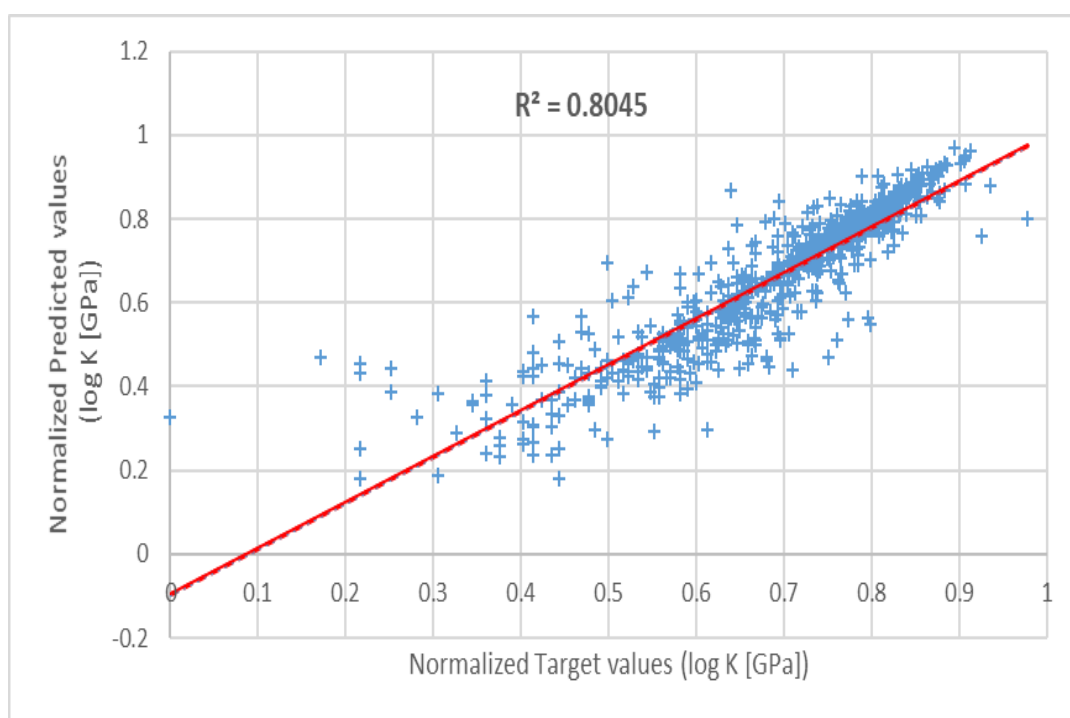


**Fig. 16 - Graph showing overlap of target and predicted values of shear modulus**

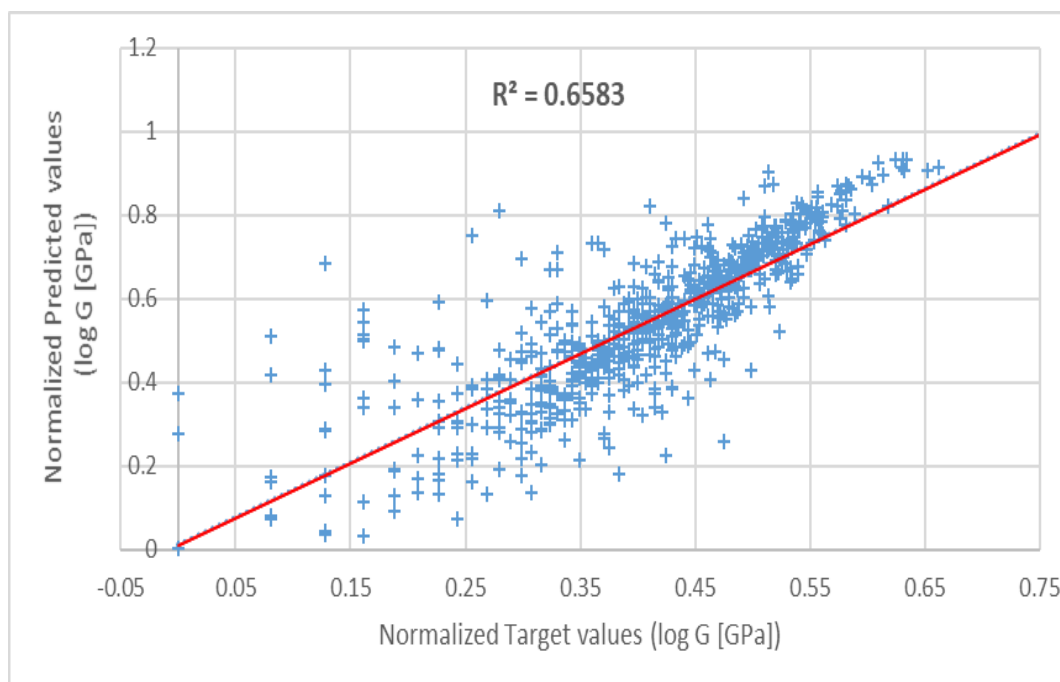
The graphs show the overlap of target and predicted values of 100 materials used to test the model whose properties have not been calculated experimentally but statistical learning based properties are available on Materials API. We can clearly infer that the RMSE values differ by a huge margin. The CGCNN performs better than the statistical models that have been used in the materials project.

3. Since bulk and shear modulus are continuous properties, we can infer the CGCNN as a regression task. Coefficient of determination ( $R^2$ ) is a useful criterion that can be used to get the measure of correlation between predicted values of bulk and shear modulus by CGCNN and the target DFT calculated values. We use normalized values of predicted and target variables to plot the scatter plots (Fig. 17, 18) for bulk and shear modulus.

Coefficient of Determination values range between 0-1 with 1 representing the highest degree of correlation between two variables and 0 representing no correlation. We obtain an  $R^2$  value of 0.8045 between predicted and target values for bulk modulus and a value of 0.6583 between predicted and target values for shear modulus. An  $R^2$  value in the range of 0.6-0.7 is generally considered a good measure of correlation between the variables. Our CGCNN model can be considered apt for prediction of material properties like bulk modulus, shear modulus, poisson's ratio, etc. since we constantly get good  $R^2$  values even on a completely unseen dataset.



**Fig. 17 – Parity plot comparing CGCNN predicted values and DFT calculated target values for bulk modulus**



**Fig. 18 – Parity plot comparing CGCNN predicted values and DFT calculated target values for shear modulus**

## 5. Future Scope

### 5.1. Porosity Informed CNN

The key features determining the effective diffusivity of a porous structure are extracted through the convolutional layers and these features are mostly connected with the input images locally. Global features or features spanning large scale may not be effectively extracted using the CNN, therefore, it may be useful to directly introduce physical parameters describing these features into the CNN model to improve its performance.

### 5.2. Pre-processed pore structures as input

CNNs exhibits relatively large error for porous samples with very small diffusivity ( $De < 0.1$ ). This is closely related to the more complex transport behaviour in porous media with very small diffusivities: in these media, the diffusion pathways are tortuous and there exist many trapped pores and dead-end paths. Since CNN models may not effectively extract features of these complicated structure spanning relatively large length scales using filters with small spatial extent, they do not perform well for such porous media. Hence, we would like to explore the possibility of improving CNN prediction by processing the images of porous structures to remove dead-end and trapped pore spaces.

### 5.3. Dendritic growth suppression in Li batteries

This work not only provides machine learning based models to predict bulk and shear modulus but also provides a pathway to solve a major challenge in lithium batteries. The dendritic growth at the interface between lithium metal and electrolytes has been a major concern while designing these batteries and it has caused severe cases of short circuit and capacity loss. Our model can be used to design new electrolytes to suppress dendrite growth. The CGCNN model is a very generalized model and can be used to predict a wide range of properties. Bulk modulus, shear modulus and Poisson's ratio are the ones we are interested in.

$$\chi = -\frac{\gamma k^2 V_M (1 + \nu)}{2z} + \frac{2G_e G_s k V_M (1 + \nu)(\nu_e(4\nu_s - 3) - 3\nu_s + 2)}{z(G_e(\nu_e - 1)(4\nu_s - 3) + G_s(4\nu_e - 3)(\nu_s - 1))} + \frac{k V_M (1 - \nu)(G_e^2(4\nu_s - 3) + G_s^2(3 - 4\nu_e))}{2z(G_e(\nu_e - 1)(4\nu_s - 3) + G_s(4\nu_e - 3)(\nu_s - 1))}$$

surface tension                      deviatoric stress                      hydrostatic stress

The stability parameter ( $\chi$ )<sup>[8]</sup> is used to determine the extent of dendritic growth has bulk modulus, shear modulus, poisson's ratio and molar volumes as the inputs. The partial molar volume of the metal in the electrolyte  $V_{Mz+}$  can be measured directly in an experiment on the difference of potentials between a stressed and unstressed electrolyte as done by Pannikkat and Raj<sup>[10]</sup> and then using the relationship  $V_{Mz+} = \partial\mu_{Mz+}/\partial p$ , where  $\mu_{Mz+}$  is the electrochemical potential of the metal ion.

## 6. References

1. Predicting Effective Diffusivity of Porous Media from Images by Deep Learning (nature.com)
2. Towards the digitalization of porous energy materials: evolution of digital approaches for microstructural design - Energy & Environmental Science (RSC Publishing) DOI:10.1039/D1EE00398D
3. Wu, J., Yin, X. & Xiao, H. Seeing permeability from images: fast prediction with convolutional neural networks. *Science Bulletin* **63**, 1215–1222 (2018)
4. Code reference - aurorlhc/property\_test: Predicting Young's Modulus of porous materials using CNN (github.com)
5. Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR
6. Xie, T.; Grossman, J. C. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Phys. Rev. Lett.* 2018, 120, 145301.
7. Ahmad, Zeeshan & Xie, Tian & Maheshwari, Chinmay & Grossman, Jeffrey & Viswanathan, Venkatasubramanian. (2018). Machine Learning Enabled Computational Screening of Inorganic Solid Electrolytes for Dendrite Suppression with Li Metal Anode. *ACS Central Science*. 4. 10.1021/acscentsci.8b00229.
8. Monroe, C.; Newman, J. The Effect of Interfacial Deformation on Electrodeposition Kinetics. *J. Electrochem. Soc.* 2004, 151, A880–A886.
9. Mathematical model of the dendritic growth during lithium electrodeposition - Rohan Akolkar\*. Department of Chemical Engineering, Case Western Reserve University,
10. Pannikkat, A.; Raj, R. Measurement of an electrical potential induced by normal stress applied to the interface of an ionic material at elevated temperatures. *Acta Mater.* 1999, 47, 3423–3431.
11. A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, et al., *Apl Materials* 1, 011002 (2013). [Materials Project Link]
12. Code reference - <https://github.com/txie-93/cgcnn>
13. [https://www.ccdc.cam.ac.uk/Community/depositastructure/cifsyntax/moreinformationaboutcifsyntax\\_v1.pdf](https://www.ccdc.cam.ac.uk/Community/depositastructure/cifsyntax/moreinformationaboutcifsyntax_v1.pdf). A short guide to Crystallographic Information Files.
14. Convolutional neural network based on SMILES representation of compounds for detecting chemical motif Maya Hirohara<sup>1</sup>, Yutaka Saito<sup>2,3</sup>, Yuki Koda<sup>1</sup>, Kengo Sato<sup>1</sup> and Yasubumi Sakakibara<sup>1</sup>
15. Density functional theory predictions of the mechanical properties of crystalline materials - CrystEngComm (RSC Publishing) DOI:10.1039/D1CE00453K

## APPENDIX

### 1. Hyperparameter tuning to predict effective diffusivity

Learning Rate	Kernel size	Stride	Pool size	Padding	No. of filters in 2 layers	Train RE	Val RE	Test RE
0.01	2x2	2x2	2x2	Same	8,16	48.43	44.00	48.44
0.01	4x4	4x4	2x2	Same	16,32	50.29	46.98	49.52
0.001	3x3	3x3	2x2	Valid	8,16	49.32	48.52	45.46
0.0001	4x4	2x2	3x3	Valid	32,64	47.31	92.66	97.01

**Table 10 – Effect of changing hyperparameters on train, validation & test relative errors**

The results for 900 valid combinations of hyperparameters has been summarized in [Results\\_Linear](#) and [Results\\_ReLU](#). All the models are trained using the same train, validation and test dataset. Hyperparameter tuning is one of the most important aspects of developing a neural network or any machine learning model. Hyperparameters are cross validated using validation dataset. The combination of hyperparameters that gives us the lowest relative error on the validation dataset is considered to be the best model. The best results for our dataset have been summarized in Table 3 – Table 6 above.

### 2. Crystallographic Information File (CIF) of LiGdAu<sub>2</sub>

```
data_LiGdAu2
_symmetry_space_group_name_H-M 'P 1'
_cell_length_a 6.91600800
_cell_length_b 6.91600800
_cell_length_c 6.91600800
_cell_angle_alpha 90.00000000
_cell_angle_beta 90.00000000
_cell_angle_gamma 90.00000000
_symmetry_Int_Tables_number 1
_chemical_formula_structural LiGdAu2
_chemical_formula_sum 'Li4 Gd4 Au8'
_cell_volume 330.80073124
_cell_formula_units_Z 4
loop_
_symmetry_equiv_pos_site_id
_symmetry_equiv_pos_as_xyz
1 'x, y, z'
loop_
_atom_site_type_symbol
_atom_site_label
_atom_site_symmetry_multiplicity
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_occupancy
```

Li	Li0	1	0.00000000	0.00000000	0.00000000	1
Li	Li1	1	0.00000000	0.50000000	0.50000000	1
Li	Li2	1	0.50000000	0.00000000	0.50000000	1
Li	Li3	1	0.50000000	0.50000000	0.00000000	1
Gd	Gd4	1	0.50000000	0.00000000	0.00000000	1
Gd	Gd5	1	0.50000000	0.50000000	0.50000000	1
Gd	Gd6	1	0.00000000	0.00000000	0.50000000	1
Gd	Gd7	1	0.00000000	0.50000000	0.00000000	1
Au	Au8	1	0.25000000	0.75000000	0.75000000	1
Au	Au9	1	0.25000000	0.25000000	0.75000000	1
Au	Au10	1	0.25000000	0.25000000	0.25000000	1
Au	Au11	1	0.25000000	0.75000000	0.25000000	1
Au	Au12	1	0.75000000	0.75000000	0.25000000	1
Au	Au13	1	0.75000000	0.25000000	0.25000000	1
Au	Au14	1	0.75000000	0.25000000	0.75000000	1
Au	Au15	1	0.75000000	0.75000000	0.75000000	1

### 3. CGCNN model summary with 4 convolutional layers and default parameters

```

CrystalGraphConvNet(
  (embedding): Linear(in_features=92, out_features=64, bias=True)
  (convs): ModuleList(
    (0): ConvLayer((fc_full): Linear(in_features=169,
out_features=128, bias=True)
      (sigmoid): Sigmoid()
      (softplus1): Softplus(beta=1, threshold=20)
      (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
      (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
      (softplus2): Softplus(beta=1, threshold=20)
    )
    (1): ConvLayer(
      (fc_full): Linear(in_features=169, out_features=128,
bias=True)
      (sigmoid): Sigmoid()
      (softplus1): Softplus(beta=1, threshold=20)
      (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
      (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
      (softplus2): Softplus(beta=1, threshold=20)
    )
    (2): ConvLayer(
      (fc_full): Linear(in_features=169, out_features=128,
bias=True)
      (sigmoid): Sigmoid()
      (softplus1): Softplus(beta=1, threshold=20)
      (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
      (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
      (softplus2): Softplus(beta=1, threshold=20)
    )
    (3): ConvLayer(

```

```

        (fc_full): Linear(in_features=169, out_features=128,
bias=True)
        (sigmoid): Sigmoid()
        (softplus1): Softplus(beta=1, threshold=20)
        (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
        (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)
        (softplus2): Softplus(beta=1, threshold=20)
    )
)
(conv_to_fc): Linear(in_features=64, out_features=128,
bias=True)
(conv_to_fc_softplus): Softplus(beta=1, threshold=20)
(fc_out): Linear(in_features=128, out_features=1, bias=True)
)

```

#### 4. Hyperparameter tuning to predict bulk modulus and shear modulus

n_conv	n_h	a_fea_len	h_fea_len	LR	Optim	MAE(b)	MAE(s)
2	1	32	64	0.01	SGD	0.097	0.142
3	1	64	128	0.01	SGD	0.099	0.145
3	2	64	128	0.01	SGD	0.101	0.145
4	1	16	32	0.01	Adam	0.103	0.153
5	2	128	512	0.01	SGD	0.276	0.298
<b>3</b>	<b>1</b>	<b>128</b>	<b>32</b>	<b>0.01</b>	<b>Adam</b>	<b>0.093</b>	<b>0.149</b>
4	1	16	32	0.001	SGD	0.11	0.153
5	2	64	128	0.001	SGD	0.120	0.156

Table 11 – Different combinations of hyperparameters trained on CGCNN