CGCNN Procedure

This dataset basically contains the properties of the crystal. Each atom is node and each edge is edge of the graph. In the neural network input. 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.

[redacted] Each node i is represented by a feature vector $\boldsymbol{v}_i$, encoding the property of the atom corresponding to node i. Similarly, each edge $(i,j)_k$ is represented by a feature vector $\boldsymbol{u}_{(i,j)_k}$ corresponding to the kth bond connecting atom i and atom j.

The convolutional neural networks built on top of the crystal graph consist of two major components: convolutional layers and pooling layers. [redacted]

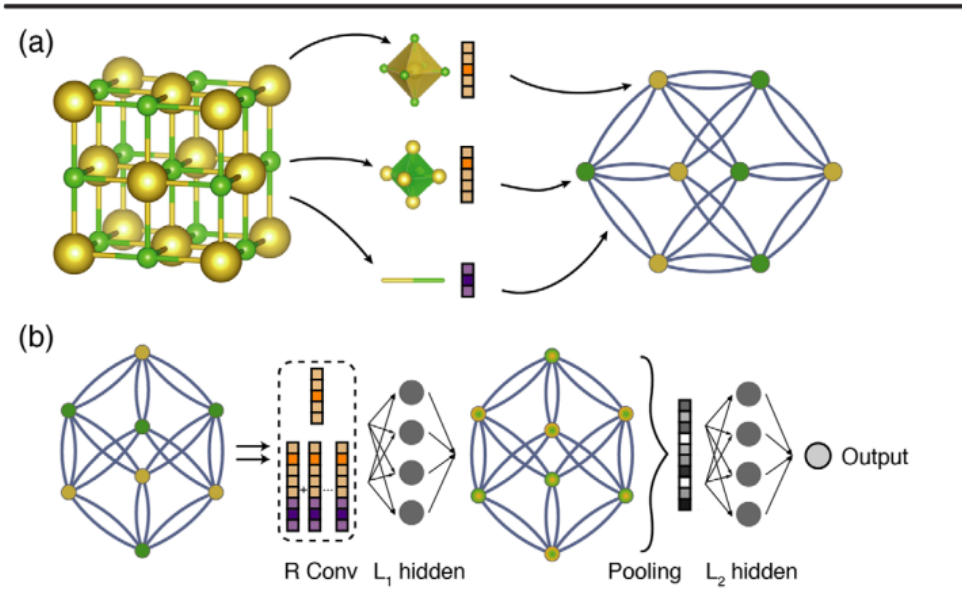

FIG. 1. Illustration of the crystal graph convolutional neural networks. (a) Construction of the crystal graph. Crystals are converted to graphs with nodes representing atoms in the unit cell and edges representing atom connections. Nodes and edges are characterized by vectors corresponding to the atoms and bonds in the crystal, respectively. (b) Structure of the convolutional neural network on top of the crystal graph. R convolutional layers and L_1 hidden layers are built on top of each node, resulting in a new graph with each node representing the local environment of each atom. After pooling, a vector representing the entire crystal is connected to L_2 hidden layers, followed by the output layer to provide the prediction.

prediction to the best of our knowledge. The convolutional layers iteratively update the atom feature vector $\boldsymbol{v}_i$ by "convolution" with surrounding atoms and bonds with a nonlinear graph convolution function,

$$\boldsymbol{v}_i^{(t+1)} = \mathrm{Conv}\left(\boldsymbol{v}_i^{(t)}, \boldsymbol{v}_j^{(t)}, \boldsymbol{u}_{(i,j)_k}\right), \quad (i,j)_k \in \mathcal{G}. \qquad (1)$$

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

$$\boldsymbol{v}_c = \mathrm{Pool}(\boldsymbol{v}_0^{(0)}, \boldsymbol{v}_1^{(0)}, \ldots, \boldsymbol{v}_N^{(0)}, \ldots, \boldsymbol{v}_N^{(R)}) \qquad (2)$$

that satisfies permutational invariance with respect to atom indexing and size invariance with respect to unit cell choice. In this work, a normalized summation is used as the pooling function for simplicity, but other functions can also be used. In addition to the convolutional and pooling layers, two fully connected hidden layers with the depths 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 $\hat{y}$.

The CGCNN is a flexible framework that allows variance in the crystal graph representation, neural network architecture, and training process, resulting in different f in Eq. (3) and prediction performance. To choose the best model, we apply a train-validation scheme to optimize the prediction of formation energies of crystals. Each model is trained with 60% of the data and then validated with 20% of the data, and the best-performing model in the validation set is selected. In our study, we find that the neural network architecture, especially the form of convolution function in Eq. (1), has the largest impact on prediction performance. We start with a simple convolution function,

$$\boldsymbol{v}_i^{(t+1)} = g\left[\left(\sum_{j,k} \boldsymbol{v}_j^{(t)} \oplus \boldsymbol{u}_{(i,j)_k}\right)\boldsymbol{W}_c^{(t)} + \boldsymbol{v}_i^{(t)}\boldsymbol{W}_s^{(t)} + \boldsymbol{b}^{(t)}\right], \qquad (4)$$

where $\oplus$ denotes concatenation of atom and bond feature vectors, $\boldsymbol{W}_c^{(t)}$, $\boldsymbol{W}_s^{(t)}$, and $\boldsymbol{b}^{(t)}$ are the convolution weight matrix, self-weight matrix, and bias of the tth layer, respectively, and g is the activation function for introducing nonlinear coupling between layers. By optimizing hyper-parameters in Table S1, the lowest mean absolute error (MAE) for the validation set is 0.108 eV/atom. One limitation of Eq. (4) is that it uses a shared convolution weight matrix $\boldsymbol{W}_c^{(t)}$ for all neighbors of i, which neglects the differences of interaction strength between neighbors. To overcome this problem, we design a new convolution function that first concatenates neighbor vectors $\boldsymbol{z}_{(i,j)_k}^{(t)} = \boldsymbol{v}_i^{(t)} \oplus \boldsymbol{v}_j^{(t)} \oplus \boldsymbol{u}_{(i,j)_k}$, then perform convolution by

$$\boldsymbol{v}_i^{(t+1)} = \boldsymbol{v}_i^{(t)} + \sum_{j,k} \sigma\left(\boldsymbol{z}_{(i,j)_k}^{(t)} \boldsymbol{W}_f^{(t)} + \boldsymbol{b}_f^{(t)}\right) \odot g\left(\boldsymbol{z}_{(i,j)_k}^{(t)} \boldsymbol{W}_s^{(t)} + \boldsymbol{b}_s^{(t)}\right), \quad (5)$$

where $\odot$ denotes element-wise multiplication and σ denotes a sigmoid function. In Eq. (5), the $\sigma(\cdot)$ functions as a learned weight matrix to differentiate interactions between neighbors and adding $\boldsymbol{v}_i^{(t)}$ makes learning deeper networks easier [29]. We achieve MAE on the validation

Above were a few parts from the paper that I felt are relevant to the methodology.

In the dataset given, we need to extract the node and edge features first of all. The method for calculating the edge and node features is given.
Number of nodes of a crystal is equal to the number of atoms in that crystal. Bar connectivity columns specify the connection between the atoms. In the columns of bar connectivity named as bar_connectivity_i_1 and bar_connectivity_i_2 , i represents the edge number and 1 and 2 represent the first atom number and 2nd atom number which that edge connects. Atoms(1 and 2) are nodes of this edge.

- Total number of atoms in a crystal is equal to max(Bar_connectivity_i_j) for a given crystal. For example in the first crystal max(Bar_connectivity_i_j)=8, there are 8 atoms. Nodal connectivity of each atom is the number of times is occurs in the bar connectivity column. For the first datapoint each atom occurs thrice in the bar_connectivity so the node connectivity of each node/atom is 3.
- Total number of edges in the crystal is max(i in the bar connectivity_i_j).
  For example in the first one it is 12.
  Features extracted from each edge are 4 (length, l, m, n)
- Form a convolution layer for each atom using

  $$\boldsymbol{v}_i^{(t+1)} = \mathrm{Conv}\left(\boldsymbol{v}_i^{(t)}, \boldsymbol{v}_j^{(t)}, \boldsymbol{u}_{(i,j)_k}\right), \quad (i,j)_k \in \mathcal{G}. \quad (1)$$

  Where in this formula i is the first atom, j is the 2nd atom and u(i,j) contains the properties of the edge connecting it. For example in the first dataset first edge bar_connectivity_1_1 and bar_connectivity_1_2 signifies that first edge is between 1st and 5th atom so 1st feature vector formed is
  (nodal connectivity(1),nodal connectivity(5), edge features(1st edge formed by 1 and 5)) == (3, 3, (length of edge between 1 and 5, l, m,n))

Similarly do it for all edges. Total number of feature vectors for a given crystal would be equal to the total number of edges of the crystal. After this train the model using the methodology specified in the paper.

1. Extract the feature vectors for nodes and edges.
   NODES-number of connections of a node: found by calculating the number of times a node is listed in the bar connectivity columns
   Bar connectivity for the first entry is -
   Node: connectivity(number of times it appears)
   1: 3
   2: 3
   3: 3
   4: 3
   5: 3
   6: 3
   7: 3
   8: 3 all the nodes from 1 to 8 appear 3 times

   EDGE- two features need to be extracted for edges
   Naming of edges- in the columns of bar connectivity named as bar_connectivity_i_1 and bar_connectivity_i_2 i represents the edge number and 1 or 2 represents which node it connects
   Eg. In the first entry the column bar_connectivity_12_1 signifies edge number 12 and 1st node of the edge is 7
   column bar_connectivity_12_2 signifies edge number 12 and 2nd node of the edge is 8
   - length of edge: obtained via nodal positions (distance formula)
     To calculate the length of the edge apply distance formula between the two nodes of the edge
     Eg. edge 12 find distance between node 7 and node 8
     Nodal coordinates are given as nodal_positions_i_j
     i represents the node number
     j =1 represents x coordinate value, j =2 represents x coordinate value
     j =3 represents x coordinate value
     Eg. Nodal_positions_7_2 represents the y coordinate value of node 7
     To calculate distance between node 7 and node 8: use formula

     $$\sqrt{(x_8 - x_7)^2 + (y_8 - y_7)^2 + (z_8 - z_7)^2} = D$$

     x_8 is given by Nodal_positions_8_1
     y_8 is given by Nodal_positions_8_2
     z_8 is given by Nodal_positions_8_3 and so on the other values

- angle each edge makes with the x,y, and z-axis: direction cosines formula implemented on the nodal points (angle with x axis= x2-x1/ distance between the points)

$$l = \frac{x_8 - x_7}{D}, \; m = \frac{y_8 - y_7}{D}, \; n = \frac{z_8 - z_7}{D}$$

4 features of each edge would be extracted - length of edge, l, m, n

Contruct a CGCNN model with the above given information.

Form a graph using node and edge parameters as given above by applying CNN with R1 layers and further create the entire CGCNN model with L1,L2 Hidden layers and pooling layers as mentioned on the page 1,2,3 of the pdf named “cgcnn” attached. Do hyperparameter tuning in the hidden layer changing various functions so as to increase accuracy of model atleast above 90 percent.

2. The feature vectors extracted above i.e. node connectivity, edge length, l, m ,n and a,b,c,alpha,beta,gamma from the spreadsheet are the input feature vector for the neural network. In the dataset 2000 points data is given. Out of 2000 datapoints, 1600 points are used for training, 200 for validation and 200 for testing.
3. Y(values to be predicted via neural network) vector is Cx, Cy, Cz and nx, ny, nz. [given in spreadsheet]

Final output required-

1. Accuracies/errors of the model in predicting the given properties.
2. Function that takes input of nodal positions, bar connectivity, a,b,c,alpha,beta,gamma as input and predits the properties Cx, Cy, Cz and nx, ny, nz with the accuracy more than 90 percent.
3. Plot of Y predicted vs Y true
4. Error plot
5. Summary/ documentation of the model