AugLiChem: data augmentation library of chemical structures for machine learning

Rishikesh Magar, Yuyang Wang, Cooper Lorsung, Chen Liang, Hariharan Ramasubramanian, Peiyuan Li and Amir Barati Farimani

1 Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States of America
2 Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States of America
3 These authors contributed equally to this work.

E-mail: barati@cmu.edu

Keywords: machine learning, data augmentation, graph neural networks

Supplementary material for this article is available online

Abstract

Machine learning (ML) has demonstrated the promise for accurate and efficient property prediction of molecules and crystalline materials. To develop highly accurate ML models for chemical structure property prediction, datasets with sufficient samples are required. However, obtaining clean and sufficient data of chemical properties can be expensive and time-consuming, which greatly limits the performance of ML models. Inspired by the success of data augmentations in computer vision and natural language processing, we developed AugLiChem: the data augmentation library for chemical structures. Augmentation methods for both crystalline systems and molecules are introduced, which can be utilized for fingerprint-based ML models and graph neural networks (GNNs). We show that using our augmentation strategies significantly improves the performance of ML models, especially when using GNNs. In addition, the augmentations that we developed can be used as a direct plug-in module during training and have demonstrated the effectiveness when implemented with different GNN models through the AugLiChem library. The Python-based package for our implementation of Auglichem: Data augmentation library for chemical structures, is publicly available at: https://github.com/BaratiLab/AugLiChem.

1. Introduction

Machine learning (ML) models, especially deep neural networks [1], capable of learning representative features from data, have gathered growing attention in computational chemistry and material science [2]. Moreover, advances from areas, such as computer vision (CV) and natural language processing (NLP), have often had a synergistic effect on ML research in computational chemistry. This has led to development of graph neural networks (GNNs) based ML models capable for accurate and efficient crystalline and molecular property prediction [3–5] and fingerprint (FP) based ML models that take chemical FPs [6–10] as input and predict target properties. However, the development of the FPs requires manual design and is sometimes time-consuming to calculate, which limits the performance of ML models leading to the increased popularity of GNN based models. Recently, GNNs [11, 12] have been a prevalence in computational chemistry, as GNNs are capable of learning representations from non-Euclidean graphs of chemical structures directly without manually engineered descriptors [13, 14]. GNNs developed for crystalline systems and molecules to predict various target properties and have shown advantages over other ML methods [15–19] and remarkably high performance. However, ML models, including GNNs and FP-based models, require large amounts of data for training [20]. Additionally, the availability of large and clean data is an important prerequisite as the performance of ML models scales with the magnitude of data [21]. However, generating large-scale properties datasets for crystalline systems or molecules usually requires sophisticated and time-consuming computational or lab experiments [22]. The experimental data can also be noisy and hard
to utilize directly. Thus, the lack of data has often prevented the use of deep learning models as they suffer from lower reliability in the small data regime [23].

The issue of data availability also plagues the areas of CV and NLP. To address the issue of data availability, data augmentation techniques have been introduced in CV and NLP, which generates new instances from available training data via some transformations to increase the amount and variance of the data [24]. Such techniques have been investigated for multiple domains, like images and text, and have demonstrated the effectiveness in improving the generalization, performance, and robustness of ML models [25–28]. For images in CV, the fundamental augmentation transformations include cropping, rotation, color distortion, etc. Following the transformations, some works manually design transformations [29] or learn the desired mixture of image transformations [30–32] during training. Augmentations have also been investigated on the representation instead of directly on input, including adding noise, interpolating, and extrapolating of image representations [33, 34]. In addition, data augmentations have been widely used for texts in NLP [35], like random deletion, insertion, and swap. One common method is to replace words with their synonyms [36] or relative word embeddings [37, 38]. Other augmentations focus on manipulation of the whole sentence, such as back translation [39] and generative models [40]. Recent works have also found that simple augmentations (e.g. cropping, resizing, color distortion for images, and token masking for texts) significantly benefit representation learning through self-supervised pre-training [41–44]. Motivated by the success of data augmentation on CV and NLP, some works in the area of molecular ML investigate augmentation for string-based molecular descriptors [45, 46] (e.g. SMILES [47]) or conformational oversampling [48]. However, data augmentation is still under-explored for chemical structures, especially for graph-based ML models. Unlike images or text, crystalline and molecular systems form graphical structures and follow chemical rules, like the composition of motifs and boundary conditions. Therefore, direct leverage of existing augmentations, such as image resizing and cropping, does not work for chemical structures. This leads us to the question: can we develop data augmentation techniques for crystalline systems and molecules to boost the performance of ML models, especially GNNs?

We develop AugLiChem: data augmentation library for chemical structures, which boosts the data availability of crystals and molecules and helps improve the performance of ML models. The augmentations can be implemented as simple plug-ins to FP-based ML models as well as GNNs. The whole pipeline of AugLiChem is illustrated in figure 1(a), which first takes in the chemical structures, either crystalline or molecular, and creates augmented data instances. The augmented data is then assigned with the same label as the original instance. The original chemical instances, together with the augmented ones, are fed into ML models for property prediction. For molecules, we introduce three molecule graph augmentations: atom masking, bond deletion, and substructure removal. Since FP calculations are different between molecules and crystalline materials, molecule graph augmentations cannot be directly utilized for FP-based ML models. We consider two extra augmentation methods for molecule FPs. Figure 1(b) illustrates both the graph and FP augmentations for molecules. Similarly, for crystals, five augmentation methods are proposed: random perturbation, random rotation, swap axes, random translation, and supercell transformation, which can be direct plug-ins for both conventional ML models and graph-based GNNs as shown in figure 1(c). Experiments on various benchmarks have demonstrated the effectiveness of our data augmentations for both crystals and molecules. Especially for GNNs, augmentation significantly boosts the performance of chemical property prediction tasks. Finally, for convenient use of our data augmentation techniques, we develop an open-source library for chemical structure augmentation based on Python as part of this work. Both crystal and molecule augmentations can be implemented with a few lines of code. Different GNN models and widely-used benchmarks are also included in the package. The package is largely self-contained with tutorials describing the functionality of different package features and example codes to run them.

2. Methods

2.1. Molecule augmentations

For molecules, we introduce augmentations that can be applied when using FP-based models and GNNs. A molecule can be embedded to a unique FP, like ECFP [10] and RDKFP. We introduce two molecular FP augmentation techniques, FP Break and FP Concatenation (FP Concat for brevity), which are proposed for FP-based ML models.

In addition, for GNN models, a molecular graph $G_m = (V_m, E_m)$ is defined where nodes $V_m$ denote atoms and edges $E_m$ denote chemical bonds between atom pairs [49, 50]. For $v_i \in V_m$, its node feature consists of two components, one-hot embedding of the atomic type $a_i$ and the chirality $c_i$, namely $v_i = [a_i, c_i]$. For $e_{ij} \in E_m$, its edge feature also consists of two components, one-hot embedding of the bond type $b_{ij}^t$ and one-hot embedding of the bond direction $b_{ij}^d$, namely $e_{ij} = [b_{ij}^t, b_{ij}^d]$. Both node and edge features can be enlarged with more attributes, like valency and aromaticity. Three molecular graph augmentation
techniques, node masking, bond deletion, and substructure removal, are also introduced and can be direct plugins to various GNN models \[17, 51\].

2.1.1. Fingerprint break
A molecule is broken into multiple-level fragments through Breaking Retrosynthetically Interesting Chemical Substructures (BRICS) \[52\] decomposition, where each fragment reserves one or more functional groups from the original chemical structure. Notice that BRICS decomposition follows a tree structure such that high-level fragments can contain low-level fragments. Fragments obtained from BRICS along with the molecule are featurized through molecular FPs, like RDK topological fingerprint (RDKFP) \[53\] and extended-connectivity fingerprint (ECFP) \[10\]. We calculate the Tanimoto similarity score \[54\] between fragments and the molecule and only keep the fragments with a score greater than a threshold \(S\). This means fragments that contain important functional groups and share similar topology with the original molecule are retained. Since the property of a molecule is heavily determined by the functional groups, fragments and the molecule are assumed to share similar properties \[55, 56\]. In FP Break, each fragment is assigned with the same label as the molecule in the training set. In the validation and test set, only the original molecules are used.

2.1.2. Fingerprint concatenation
On the other hand, FP Concat follows the same BRICS decomposition as the FP Break, except that the former concatenates randomly picked \(K\) FPs from fragments and the molecule, and assigns the same label as the original molecule to each concatenated FP. Additionally, FP Concat includes a replicated FP that repeats the FP of the original molecule \(K\)-times, so that the original FP can be used with the trained models. In the
testing phase, models are tested only on the replicated FPs from the validation and test sets. In our implementation of both FP Break and FP Concat, $S$ is set as 0.6 and $K$ as 4. In atom masking, a node $v_i$ is randomly masked by replacing the node feature $a_i$ with the mask vector $m$. Bond deletion removes an edge $e_{ij}$ from the molecular graph with a certain probability. And substructure removal follows the BRICS decomposition in FP augmentations, where molecular graphs are created based on the decomposed fragments from BRICS and are assigned with the same label as the original molecule.

2.1.3. Atom masking
In atom masking, a node $v_i$ is randomly masked by replacing the node feature $a_i$ with the mask vector $m$, which is a unique vector different from all the node features in the database. Thus, the GNNs lose the atomic information concerning the masked atoms during training, which forces the model to learn the correlation between atoms through messages passing from the adjacencies.

2.1.4. Bond deletion
Bond deletion randomly removes an edge $e_{ij}$ from the molecular graph given a certain probability, so that the node pair $(v_i,v_j)$ are not directly connected within the molecular graph. This means the model is forced to aggregate information from atoms that are not adjacent in the molecular graph. Bond deletion relates to the breaking of chemical bonds which prompts the model to learn correlations between the involvements of one molecule in various reactions.

2.1.5. Substructure removal
Substructure removal follows the BRICS decomposition in FP augmentations, where molecular graphs are created based on the decomposed fragments from BRICS and are assigned with the same label as the original molecule. Fragment graphs contain one or more important functional groups from the molecule, and GNNs trained on such augmented data learn to correlate target properties with functional groups.

2.2. Crystalline augmentations
We treat a crystalline system as an undirected graph with atoms as the nodes and the bonds as the edges. The crystal graph can be denoted by $G_{cr} = (V_{cr}, E_{cr})$, where $V_{cr}$ are the nodes of the graph denoting the atom sites in a crystal structure and $E_{cr}$ represent the edges. The idea of a bond in crystalline systems is not as concretely defined as in molecular graphs. Therefore, to construct the edges between the nodes, all atoms within a radius are considered to be bonded. In general, the strength of the bond is then determined by the amount of distance from the central atom and edge features are encoded accordingly. After constructing the graph, we treat it as an input feature to the GNN model, the GNN model then learns from the graph and predicts the material property.

We implement five different types of crystal augmentations as a part of this work. The augmentations we implement include random perturbation, random rotation, swap axes, random translate, and supercell transformation. Each of the augmentations that we implement creates a different instance of the crystal enabling the ML models, particularly GNNs to take advantage of the large data set and generate accurate representations of the crystal. We show that by using augmentations we can improve the prediction performance of GNNs and shallow ML models on a majority of the benchmark data sets. We would like to note that these augmentation strategies have been applied only for the training set and not for the validation and test sets.

2.2.1. Random perturbation
In the random perturbation augmentation, all the sites in the crystalline system are randomly perturbed by a distance between 0 and 0.05 Å. This augmentation is especially useful in breaking the symmetries that exist between the sites in the crystals.

2.2.2. Random rotation
In the random rotation transform, we randomly rotate the sites in the crystal between 0 and 360 degrees. To generate the augmentation we initially use random perturbation augmentation to generate the initial structure which is then rotated randomly.

2.2.3. Swap axes
In the swap axes augmentation strategy, we swap the coordinates of the sites in the crystal between two axes. For example, we may swap the $x$ and the $y$ axes coordinates or the $y$ and $z$ axes coordinates. The lattice vectors are not swapped, which greatly displaces the locations of all the sites in the crystal. This augmentation strategy has been inspired by work done by Kim et al [57].
Figure 2. GNNs for crystalline systems and molecules. (a) A molecular graph contains nodes representing atoms, while edges are defined as chemical bonds between atoms. The built molecule graph is then fed to the GNNs and fully-connected layers to predict molecular properties. (b) In the crystal ML pipeline, we start with an input crystal. The graph representation is constructed with atoms representing the nodes $V$ and the bonds representing the edges $E$ connecting adjacent nodes. Once the graph is constructed, the graph convolutional operations (shown in green) are conducted followed by the fully-connected layers to predict the target property.

2.2.4. Random translate
The random translate transforms randomly displaces a site in the crystal by a distance randomly between 0 and 1 Å. In this work, we randomly select 25% of the sites in the crystal and displace them. This creates an augmentation different from the random perturb augmentation as not all the sites in the crystal are displaced.

2.2.5. Supercell transformation
The supercell transformation produces a supercell of the crystalline system. To apply the supercell transform we first apply the random perturbation and subsequently convert the crystal to a supercell. The distinct feature of the supercell of the crystal is that after transformation the supercell represents the same crystal with a larger volume. There exists a linear mapping between the basis vectors of crystal and the basis vectors of the supercell.

2.3. Graph neural networks for chemical structures
GNNs update node-level representations using the aggregated message passing between nodes and extract graph-level representations through readout [58]. Recently, GNNs have been widely investigated in modeling chemical structures [59]. GNNs have been successfully used on the molecular graphs to learn FPs [60, 61], predict molecular properties [15–17, 62], and generate target molecules [63–65]. Chemical structures, both organic molecules and inorganic structures, are represented as graphs. In this work, each node represents an atom [50] and edges can be defined as covalent bonds (molecules) [17, 49, 60] or connection to adjacent atoms (crystals) [14, 18]. This makes molecular graphs naturally translationally and rotationally invariant. Examples of building a graph from a molecule and a crystalline system are illustrated in figure 2.

GNNs utilize a neighborhood aggregation operation, which update the node representation iteratively. The update of each node in a GNN layer is given in equation (1):

\[
\begin{align*}
\mathbf{a}_v^{(k)} &= \text{AGGREGATE}^{(k)} (\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}), \\
\mathbf{h}_v^{(k)} &= \text{COMBINE}^{(k)} (\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}),
\end{align*}
\]

where $\mathbf{h}_v^{(k)}$ is the feature of node $v$ at the $k$-th layer and $\mathbf{h}_v^{(0)}$ is initialized by node feature $\mathbf{x}_v$. $\mathcal{N}(v)$ denotes the set of all the neighbors of node $v$. GNNs implemented in this work have multiple aggregation and combination functions, including mean or max pooling [11, 66], fully-connected (FC) layers [12], recurrent neural network (RNN) [66, 67], Gaussian weighed summation [18, 19], and attention mechanism [49, 68]. Other works exploit edge features in the message passing to leverage more information [50, 69]. To extract a
graph-level feature \( h_G \) from the node features, readout operation integrates all the node features among the graph \( G \) as given in equation (2):

\[
    h_G = \text{READOUT}(\{ h_v^{(k)} : v \in G \}).
\] (2)

Common readout operations can be summation, averaging, or max pooling over all the nodes [12]. More recently, differentiable pooling readout has been developed [70, 71]. Given \( h_G \), either FC layers [50] or RNNs [49] are developed to map the graph feature to the predicted property.

Unlike two-dimensional (2D) molecular graphs, crystal graphs usually require the 3D positional data as the input. Crystal graphs treat the atoms in the crystal as nodes and edges in crystal graphs are defined as the relationship between neighboring atoms [72, 73]. Following the setting, various aggregation operations [18, 74] and crystalline features [19, 75–77] have been investigated in predicting crystal properties using GNNs. Additionally, some works also consider 3D positional features in molecular GNNs for property predictions concerning quantum mechanics [13, 14].

In this work, we use data augmentation strategies for multiple GNN backbones for both the molecular systems and crystalline materials. For molecules, we implement GCN [11], GIN [12, 50], DeeperGCN [67, 78], and AttentiveFP [49] as GNN backbones to predict molecular properties. These models follow the message passing framework but differ in detailed aggregation and combination operations. GCN aggregates all the neighbors through mean pooling followed by a linear mapping with ReLU activation. GIN, on the other hand, sums up all the neighboring node features and develops an MLP afterward. However, such GNN models suffer from the over-smoothing and the vanishing gradient problems when the number of graph convolutional layers increases [67, 79]. Over-smoothing indicates that all the nodes have indistinguishable representations and vanishing gradient refers to that the gradient at some layers is vanishingly small causing the weights unchanged through backpropagation. Due to the issues, GNN models, like GCN and GIN, are usually limited to less than five layers. In our case, we develop GCN with three layers and GIN with five layers following previous works [17, 50]. DeeperGCN [78] introduces skip connections between layers in ResNet [80] to graph convolutional layers which greatly eliminates vanishing gradient and enables GNNs with much more layers. We implement a 28-layer DeeperGCN based on the design in the original literature. AttentiveFP utilizes the attention mechanism and RNNs to aggregate neighbor information and update node features, which reaches state-of-the-art on several challenging molecular property benchmarks. These models cover a wide variety of GNN frameworks and greatly demonstrate the effectiveness of our molecule augmentation techniques.

For the crystalline systems we use the CGCNN [18], GIN [12, 50], and GCN [11]. The CGCNN model builds a crystal graph by taking the atoms as the nodes and modeling the interaction between the atoms as the edges. To describe the edge features the CGCNN model uses Gaussian featurization. The model uses 3 graph convolutional layers to aggregate node and edge features and does the material property prediction. The GCN model aggregates features over all the neighbors through mean pooling followed by a linear mapping with ReLU activation. GIN, on the other hand, sums up all the neighboring node features and develops an MLP afterward. We use mean pooling as the readout function for the GIN and use the pooled vector to extract the representations of the crystalline system that is later used for property prediction.

### 2.4. Training details

We benchmark our augmentation methods on different crystalline and molecule datasets. For the twelve molecule benchmarks, instances are split into train/validation/test by the ratio of 8:1:1 via scaffold splitting, which provides a challenging yet realistic setting [17, 50]. The percentages of train/validation/test sets have been kept the same baseline models which are trained without augmented data and the models trained with augmented data. Augmentations are implemented on the training sets only while validation and test sets stay intact. Each benchmark is split into the same training/validation/test sets, and the same augmentations are implemented on instances on the training set if applied. So that all the ML models are trained and tested on the same instances for a fair comparison. Trained ML models are tested on validation sets to select the best performing model while the test sets are not accessed during training and validation. For each benchmark, we conduct three individual splitting and training and report the average and standard deviation of the three results on the test set.

In case of the five crystalline benchmarks, instances are first randomly split into the train set and test set for five-fold cross validation. The ratio of train set and test set is 4:1. We further split the training data into training and validation sets in the ratio of 4:1. Finally, we have 64% of the data as training data, 16% of the data as validation data, and the remaining 20% as the test set. We report the average and standard deviation of the five results on the test set, following previous works [19]. For example, the band gap benchmark containing 26 709 crystalline instances is split into train/validation/test sets and each includes
Table 1. Results of different ML models on multiple molecular classification benchmarks with and without molecular data augmentations. Both the average and standard deviation of the test ROC-AUC scores are reported. Bold indicates performance improvement.

| Dataset       | BBP     | BACE    | ClinTox | HIV     | MUV     | Tox21   | SIDER   |
|---------------|---------|---------|---------|---------|---------|---------|---------|
| # of data     | 2039    | 1513    | 1478    | 41 127  | 93 087  | 7831    | 1427    |
| # of tasks    | 1       | 1       | 2       | 1       | 17      | 12      | 27      |

| Models        | ROC-AUC on test set (scaffold split) |
|---------------|-------------------------------------|
| RF            | 53.34 (1.19) 71.11 (0.46) 50.00 (0.00) 54.59 (1.43) 50.00 (0.00) 53.19 (0.49) 52.13 (1.27) |
| RF\textsuperscript{aug} | 54.29 (0.45) 74.87 (1.83) 50.83 (1.18) 55.48 (0.19) 50.62 (0.00) 54.68 (0.78) 53.80 (1.42) |
| SVM           | 58.12 (0.00) 73.35 (0.00) 60.35 (0.00) 58.75 (0.00) 49.98 (0.00) 55.78 (0.00) 56.00 (0.00) |
| SVM\textsuperscript{aug} | 58.62 (0.00) 76.70 (0.00) 66.49 (0.00) 62.73 (0.00) 53.74 (0.00) 57.83 (0.00) 56.52 (0.00) |
| GCN           | 71.82 (0.94) 75.63 (1.95) 67.42 (7.62) 74.05 (3.03) 71.64 (4.00) 70.86 (2.58) 53.60 (3.21) |
| GCN\textsuperscript{aug} | 71.66 (1.13) 82.32 (1.87) 75.20 (2.60) 75.76 (0.98) 84.68 (3.63) 74.63 (1.47) 68.99 (2.42) |
| GIN           | 71.69 (1.08) 53.66 (5.98) 61.40 (1.22) 68.48 (1.51) 67.99 (5.70) 70.24 (1.72) 58.84 (4.19) |
| GIN\textsuperscript{aug} | 73.64 (1.17) 80.27 (2.26) 76.81 (6.29) 77.31 (0.32) 80.01 (3.34) 75.15 (1.29) 67.97 (2.01) |
| DeeperGCN     | 61.01 (3.92) 75.73 (5.29) 57.56 (1.40) 65.67 (1.71) 67.99 (5.70) 70.78 (1.54) 53.60 (3.34) |
| DeeperGCN\textsuperscript{aug} | 70.00 (0.51) 86.51 (0.39) 64.23 (3.06) 75.00 (0.75) 81.01 (5.30) 74.80 (1.13) 65.45 (1.69) |
| AttentiveFP   | 68.56 (1.16) 81.39 (1.00) 69.97 (4.53) 75.46 (2.32) 63.60 (7.83) 70.04 (2.63) 57.57 (4.13) |
| AttentiveFP\textsuperscript{aug} | 73.58 (0.65) 83.32 (0.53) 76.07 (5.98) 77.96 (1.27) 78.22 (3.06) 75.89 (1.31) 67.90 (2.16) |

17 093/4274/5342 instances. If five augmented instances are created for each data point in the band gap training data, we get 85 465 augmented instances in addition to 17 093 instances in the original training set, which adds up to a total of 102 558 instances in the augmented training set.

3. Results and discussions

3.1. Investigation of molecule augmentations

We consider two molecular FP augmentations for FP-based ML models and three molecular graph augmentations [17, 51] for GNNs. As shown in table 1, Random Forest (RF) and Support Vector Machine (SVM) are implemented on top of RDKFP [53] and ECFP [10] for 7 classification benchmarks from MoleculeNet [4]. All the benchmarks are split through scaffold splitting to provide a more challenging yet realistic way of splitting [50]. Within each benchmark, a model is trained for three individual runs, and both the mean and standard deviation of compute areas under the receiver operating characteristic curve (ROC-AUC) on test sets are reported. All the experimental results for molecules are reported following this setting unless further mentioned. Models with an ‘aug’ subscript indicate models are trained with an augmented data set. In almost all the classification benchmarks, both RF and SVM trained with augmented data sets outperform baseline models trained only on original data sets. For instance, RF augmentations improve the ROC-AUC of SVM by more 6% on ClinTox and by 3% on BACE. It is demonstrated that assigning the same label for FP augmented data enlarges the available training set and sharpens the decision boundary of ML models. Also, performances of FP augmentations on 5 regression benchmarks: FreeSolv, ESOL, Lipo, QM7, and QM8, are investigated as shown in table 2. Similar to classifications, all benchmarks are scaffold split. The mean and standard deviation of root mean square errors (RMSEs) in three individual runs are reported. In 4 out of 5 regression benchmarks, augmentations help boost the performance of SVM models. However, RFs trained with augmented data fail to improve the RMSE in comparison to baseline results except on ESOL benchmark. This could be because unlike classification tasks where labels are usually abstract and sparse, labels for regression are more sensitive, and shallow ML models trained on augmented data can be distracted from the ground truth labels. More detailed results of RF and SVM can be found in supplementary information (tables S1 and S2).

Besides FP augmentation, we introduce three molecular graph augmentation techniques, node masking, bond deletion, and substructure removal, focusing on GNN models [17, 51]. To demonstrate how molecular graph augmentation benefits the molecular property prediction, we implement multiple GNN models, including GCN [11], GIN [12, 50], DeeperGCN [67, 78], and AttentiveFP [49], and compare the results trained with and without augmentations. For more details of the GNN implementation and training and benchmark datasets used for molecules, please refer to supplementary information (sections D and F). Table 1 shows the test ROC-AUC of GNNs model on multiple classification benchmarks. It is demonstrated that on all the 7 classification benchmarks containing 61 tasks, GNN models trained with augmented data surpass baseline results by significant amounts. For example, GIN models trained with augmentations achieve an average 11.27% improvement compared to GIN with no augmentation. Also, molecular graph
augmentation improves the test RMSE results on all the regression benchmarks as shown in table 2. On FreeSolv data set, a small yet challenging benchmark concerning molecular hydrogen free energy in water, augmentations reduce RMSE by 0.575 in the average of the four GNN models, which is a 20.55% improvement with respect to the non-augmented baseline models. Such augmentation techniques not only benefit training with limited data, but also improve performance on large data sets with tens of thousands of molecules, like HIV, MUV, and QM8. Results shown in tables 1 and 2 demonstrate that our molecular graph augmentations improve the performance of challenging classification and regression property prediction tasks on various GNN models, regardless of different message passing and aggregation functions.

### 3.2. Investigation of crystalline augmentations

To improve the performance of ML models, especially GNNs, we introduce five data augmentation strategies that enhance the amount of data available for training, namely random perturbation, rotation, random swap axes, random translation, and supercell transformation. To evaluate the effects of using these augmentations when training GNN models, we benchmark the performance of the different models with data augmentation on six different datasets and observe performance gains in all the cases. The six datasets that we include in our study are: band gap [81], fermi energy [81], formation energy [81], lanthanides [8], perovskites [82], and HOIP [83]. The average and standard deviation of test mean absolute error (MAE) on crystalline benchmarks are demonstrated in table 3. The final augmentation strategy was determined by training and evaluating each model on each individual transformation first and evaluating their performance. The best two augmentations with the lowest MAE were then used for additional training and evaluation. The best results from either the combination or single augmentation strategy are reported. We found two augmentations tended to perform best while also being faster than using more augmentations. The results using a single augmentation strategy are shown in figure 5.

The models trained on the augmented data are denoted with subscript 'aug'. For example, the CGCNN model trained on the augmented data is denoted by CGCNN\textsubscript{aug}. We observe performance gains on all the data sets except Fermi Energy and GCN on HOIP for all the deep models validating the usefulness of data set augmentations to improve the performance of the models. We observe a gain between 10% and 33% for the five data sets with CGCNN. GCN shows improvement between 2% and 12%, and GIN model shows improvement between 2% and 33%. These improvements indicate the gain in performance of models when trained with augmentations. Additional information on crystalline datasets and training details for the GNN model are available in the supplementary information (sections E and G). We also use the augmentations with shallow ML algorithms and examine the performance of these models on the five data sets. To facilitate the crystalline systems, we use the AGNI FPs from Botu et al [9]. The AGNI FP is a vector of size 32 that represents the crystalline systems. The vector can then be used as a feature for conventional ML algorithms like Random Forest and SVM. The results obtained for SVM and RF, however, do not show significant improvement for all the data sets as in the case of GNN algorithms. We can conclude empirically the effects of data augmentation are more pronounced for deep learning-based architectures than the conventional ML algorithms.

| Dataset | FreeSolv | ESOL | Lipo | QM7 | QM8 |
|---------|----------|------|------|-----|-----|
| # of data | 642 | 1128 | 4200 | 6830 | 21786 |
| # of tasks | 1 | 1 | 1 | 1 | 12 |
| Units kcal mol\textsuperscript{-1} | log (mol L\textsuperscript{-1}) | N/A | kcal mol\textsuperscript{-1} | eV |
| Models | RF\textsubscript{aug} | SVM\textsubscript{aug} | GIN\textsubscript{aug} | DeeperGCN\textsubscript{aug} | AttentiveFP\textsubscript{aug} |
| RMSE on test set (scaffold split) | 4.140 (0.007) | 3.092 (0.000) | 2.368 (0.106) | 2.292 (0.104) | 1.779 (0.081) |
| | 1.528 (0.021) | 1.433 (0.000) | 1.431 (0.057) | 1.358 (0.039) | 0.857 (0.016) |
| | 1.235 (0.028) | 0.8148 (0.0000) | 0.8095 (0.0102) | 0.8186 (0.0000) | 0.7518 (0.0092) |
| | 0.7914 (0.0310) | 120.8 (1.8) | 128.1 (2.9) | 124.9 (1.1) | 117.8 (1.6) |
| | 118.6 (1.1) | 0.03 480 (0.00 081) | 0.04 244 (0.00 115) | 0.03 466 (0.00 060) | 0.03 419 (0.00 081) |
| | 0.03 677 (0.00 040) | 0.03 708 (0.00 092) | 0.03 708 (0.00 092) | 0.03 708 (0.00 092) | 0.03 419 (0.00 081) |
Table 3. Results of different GNN models on benchmark datasets with and without crystalline systems data augmentation. Only the data points in the training set have been augmented. Both the average and standard deviation of the MAEs are reported. Bold indicates performance improvement.

| Dataset      | Band gap | Fermi energy | Formation energy | Lanthanides | Perovskites | HOIP |
|--------------|----------|--------------|------------------|-------------|-------------|------|
| # of data    | 26 709   | 27 779       | 26 078           | 41 333      | 18 928      | 1346 |
| Units        | eV       | eV           | eV atom⁻¹        | eV atom⁻¹  | eV atom⁻¹  | eV   |

| Models       | Mean absolute error (MAE) on test set (random split) |
|--------------|-------------------------------------------------------|
| RF           | 0.928 (0.006) 1.128 (0.012) 0.447 (0.008) 0.468 (0.01) 0.486 (0.004) 0.256 (0.011) |
| RFaug        | 0.938 (0.009) 1.142 (0.011) 0.452 (0.006) 0.460 (0.01) 0.482 (0.004) 0.251 (0.014) |
| SVM          | 1.052 (0.008) 1.248 (0.012) 0.517 (0.006) 0.889 (0.021) 0.528 (0.005) 0.319 (0.015) |
| SVMaug       | 1.047 (0.008) 1.267 (0.012) 0.523 (0.006) 0.869 (0.02) 0.522 (0.004) 0.312 (0.015) |
| CGCNN        | 0.276 (0.009) 0.205 (0.008) 0.031 (0.001) 0.067 (0.003) 0.083 (0.006) 0.199 (0.015) |
| CGCNNaug     | 0.243 (0.009) 0.210 (0.006) 0.028 (0.001) 0.042 (0.002) 0.069 (0.002) 0.183 (0.011) |
| GIN          | 0.524 (0.007) 0.767 (0.068) 0.079 (0.002) 0.206 (0.034) 0.363 (0.003) 0.303 (0.022) |
| GINaug       | 0.486 (0.007) 0.697 (0.011) 0.073 (0.001) 0.137 (0.008) 0.356 (0.002) 0.287 (0.012) |
| GCN          | 0.508 (0.004) 0.704 (0.019) 0.083 (.002) 0.142 (0.006) 0.370 (0.006) 0.299 (0.022) |
| GCNaug       | 0.497 (0.003) 0.704 (0.013) 0.077 (0.002) 0.125 (0.004) 0.362 (0.005) 0.297 (0.012) |

Figure 3. Using the CGCNN model, we see the augmented training data has much better coverage of the t-SNE embedding space. Better coverage by the training set means fewer test points lie outside of the training distribution, leading to better performance.

Additionally, to investigate why augmentation strategies were effective for some datasets and not for others. We explored the latent representation of a trained model using t-SNE [84] embedding. The embedding was done using the output from the final graph layers and training t-SNE to get two features. We see in figure 3 that the yellow training data covers significantly more of the black test points, especially for the five data sets in which we see improved results. This larger coverage may help models learn a more generalized representation of the data, leading to better results. This is especially clear for the Band Gap dataset. In the unaugmented embedding, we see many black test points on not covered by the yellow training points, but with the augmented data, we see the yellow training points cover the black test points almost entirely. We see a corresponding improvement in performance, where the model trained on augmented data has performance improved by 12%.

3.3. Combination of different graph augmentations
To understand the effect of augmentations and evaluate the best strategy for using these augmentations, we conduct a systematic ablation study for both crystals and molecules. To investigate the optimal augmentation strategy in molecules, we compare the performance of different molecular graph augmentations as well as their combinations. Figure 4 shows the test results, where the mean and standard deviation of 3 individual runs are illustrated by the height of each column and error bar respectively. In particular, 6 augmentation strategies are considered: no augmentation (None), node masking (NM), bond deletion (BD), substructure removal (SR), node masking with bond deletion (NM+BD), and all augmentations combined (NM+BD+SR). For comparison, AttentiveFP models with the same settings are implemented for each benchmark with different augmentations. More implementation details and results concerning augmentation combinations can be found in supplementary information (tables S3–S5).
Figure 4. Comparison of molecular graph augmentations and their combinations. (a) Test ROC-AUC on classification benchmarks. (b) Test normalized RMSE on regression benchmarks.

Test ROC-AUC of classification tasks is illustrated in figure 4(a). Augmentations, regardless of the specific strategies, improve performance on most benchmarks. When applying one augmentation solely, NM and BD are generally better choices compared to the SR. Further, the combination of NM and BD shows rival improvement as NM or BD, and even surpasses the single augmentation strategy in multiple benchmarks, including BBBP, HIV, Tox21, and SIDER. Figure 4(b) demonstrates the test normalized RMSE of different augmentations on regression tasks. Normalized RMSE is calculated by dividing the RMSE by the range of labels in each dataset. BD achieves the best performance among NM, BD, and SR in most benchmarks. Results from NM are also close to BD and even better in Lipo dataset. Similarly, NM with BD obtains competitive performance and surpasses other strategies in three out of all five benchmarks. In general, NM, BD, and the combination of these two are the best performing augmentation strategies in most classification and regression benchmarks. However, it should be pointed out that the optimal augmentation strategy is task-dependent. For instance, though SR struggles to compete with other augmentations in most cases, in BACE and MUV, SR still achieves comparable improvement to NM and BD. Also, the combination of all three molecular graph augmentations does not perform well compared to other strategies. This may be due to such combinations generating augmented graphs that are hard to match with the original graph.

For crystalline systems, we trained using all the five augmentations and evaluate the performance of CGCNN model using MAE as a performance metric. The plot for the normalized MAE for all the five datasets with different augmentation strategies has been shown in figure 5. To find the optimal augmentation strategy we experiment with different augmentation strategies: no augmentation (None), random perturbation (RP), random rotation (RR), supercell transformation (ST), swap axes transform (SW), and random translation (RT). To conduct the ablation all the training parameters were kept the same to ensure fair comparison among the augmentation strategies. We observe that the optimal augmentation strategy is task-specific and data-dependent, we select the two best performing augmentation strategies on the dataset. The final results are then reported as the best performing single or double augmentation strategy. For example, if we consider the perovskites dataset with CGCNN as the GNN model, the random perturbation and supercell transform are the best performing augmentation strategies with the lowest MAE. For the CGCNN\textsubscript{aug} model we train on dataset which is an aggregated dataset from the random perturbation, supercell transformation and the original crystals perovskites dataset. Thus, we get three time more training data using data augmentation strategies proposed for crystalline materials. However, for GCN on HOIP, the random perturbation and swap axes transforms had the best results, but random perturbation alone had
better results than their combination, so that was chosen as the best augmentation strategy. The best augmentation strategies for all GCN models are available in supplementary information (table S6).

3.4. AugLiChem package
The package is designed with ease of use in mind for all aspects. Publicly available data is downloaded and cleaned automatically, significantly reducing the time to start running experiments. Implementations of popular models are available for easy comparison. Both random splitting and k-fold cross-validation can be done for crystal data sets. Single and multi-target training can be done with molecule data sets, where handling missing data is done automatically. These training styles are included for ease of experimentation. Both molecule and crystal data handling are also designed so only the training set is augmented, ensuring evaluation validity. Examples can be found in supplementary information (section H). Detailed usage guides and documentation for all relevant functions and models: [https://baratilab.github.io/AugLiChem/](https://baratilab.github.io/AugLiChem/).

4. Conclusion
The work introduces a systematic methodology for augmentations of molecular data sets. We design augmentation strategies for crystalline systems and molecules. Using these augmentations, we are able to significantly boost the amount of data available for the ML models. For crystalline systems, we design five different augmentation strategies and select the most effective ones to train ML models. Using the augmented data we observe significant gains in the performance of three different GNNs on six different data sets consisting of a wide range of properties of crystalline systems. Similarly, for molecules, we investigate two augmentation strategies for FP-based ML and three strategies for GNN models. We observe performance gains on various popular benchmarks (seven classification and five regression data sets in total) when compared to the performance of baseline GNNs without augmentation, indicating the effectiveness of augmentation strategies. The performance gains observed when training GNN models empirically prove the effectiveness of data augmentation strategies on crystalline and molecular data sets. Finally, we also develop an open-source python package that implements all the augmentations introduced in the paper and can be imported directly into any ML workflow.

Data availability statement
The data that support the findings of this study are openly available in Materials Project (Reference Number: [79]), Perovskites dataset (Reference Number: [80]), HOIP dataset (Reference Number: [81]), Lanthanides dataset (Reference Number: [8]), MoleculeNet Benchmark (Reference Number: [4]).

The data that support the findings of this study are openly available at the following URL/DOI: [https://github.com/BaratiLab/AugLiChem/](https://github.com/BaratiLab/AugLiChem/).

Acknowledgment
The work was supported by the start up fund from the Mechanical Engineering Department at Carnegie Mellon University.
ORCID iDs

Rishikesh Magar  https://orcid.org/0000-0001-6216-0518
Yuyang Wang  https://orcid.org/0000-0003-0723-6246
Cooper Lorsung  https://orcid.org/0000-0003-1410-9077
Amir Barati Farimani  https://orcid.org/0000-0002-2952-8576

References

[1] LeCun Y, Bengio Y and Hinton G 2015 Deep learning Nature 521 436–44
[2] Keith J A, Vassilev-Galindo V, Cheng B, Chmiela S, Gasteiger M, Müller K-R and Tkatchenko A 2021 Combining machine learning and computational chemistry for predictive insights into chemical systems Chem. Rev. 121 9816–72
[3] Fung V, Zhang J, Juarez E and Sumpter B G 2021 Benchmarking graph neural networks for materials chemistry npj Comput. Mater. 7 1–8
[4] Wu Z, Ramsundar B, Feinberg E N, Gomes I, Geniesse C, Pappu A S, Leswing K and Pande V 2018 MoleculeNet: a benchmark for molecular machine learning Chem. Sci. 9 513–30
[5] Elton D C, Boukouvalas Z, Fuge M D and Chung P W 2019 Deep learning for molecular design—a review of the state of the art Mol. Syst. Des. Eng. 4 828–49
[6] Bartók A P, Kondor R and Csányi G 2013 Phys. Rev. B 87 184115
[7] Hoo H and Rupp M 2017 Unified representation of molecules and crystals for machine learning (arXiv:1704.06439)
[8] Lam Pham T, Knoh K, Terakura K, Miyake T, Tsuda K, Takigawa I and Chi Dam H 2017 Machine learning reveals orbital interaction in materials Sci. Technol. Adv. Mater. 18 756–69
[9] Botu V, Batra R, Chapman J and Ramprasad R 2017 Machine learning force fields: construction, validation and outlook J. Phys. Chem. C 121 511–22
[10] Rogers D and Hahn M 2015 Fast subtree kernels for contextualized word embeddings J. Mach. Learn. Res. 16 3205–40
[11] Yang K, Swanson K, Jin W, Coley C, Eiden P, Gao H, Guzman-Perez A, Hopper T, Kelley B and Mathea M et al 2019 Analyzing learned molecular representations for property prediction J. Chem. Inf. Model. 59 3370–88
[12] Wang Y, Wang J, Cao Z and Barati Farimani A 2020 Orbital graph convolutional neural network for material property prediction Sci. Rep. 10 1–10
[13] Wang Y, Cao Z and Barati Farimani A 2021 Molecular contrastive learning of representations via graph neural networks Nat. Mach. Intell. 4 279–87
[14] Xu T and Grossman J C 2018 Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties Phys. Rev. Lett. 120 145301
[15] Karamad M, Magar R, Shi Y, Siahrostami S, Gates I D and Farimani A B 2020 Orbital graph convolutional neural network for material property prediction Phys. Rev. Mater. 4 093801
[16] Hestness J, Narang S, Ardalani N, Diamos G, Jun H, Kainjevid J, Patwary M, Ali M, Yang Y and Zhou Y 2017 Deep learning scaling is predictable, empirically (arXiv:1712.00409)
[17] Zhu X, Vondrick C, Fowlkes C C and Ramanan D 2016 Do we need more training data? Int. J. Comput. Vis. 119 76–92
[18] Brown N, Fisato M, Segler M H S and Vaucher A C 2022 GuacaMol: benchmarking models for de novo molecular design J. Chem. Inf. Model. 59 1090–108
[19] Bengio Y, Lecun Y and Hinton G 2021 Deep learning for AI Commun. ACM 64 58–65
[20] Botton J and Papakipos Z 2021 AugLy: a data augmentations library for audio, image, text, and video (available at: https://github.com/facebookresearch/AugLy)
[21] Shorten C and Khoshgoftaar T M 2019 A survey on image data augmentation for deep learning J. Big Data 6 1–48
[22] Kobayashi S 2018 Contextual augmentation: data augmentation by words with paradigmatic relations (arXiv:1805.06201)
[23] Coulombe C 2018 Text data augmentation made simple by leveraging NLP cloud APIs (arXiv:1812.04718)
[24] Chen T, Kornblith S, Norouzi M and Hinton G 2020 A simple framework for contrastive learning of visual representations Proc. 37th Int. Conf. on Machine Learning (ICML'17) vol 119 pp 1257–67
[25] DeVries T and Taylor G W 2017 Improved regularization of convolutional neural networks with cutout (arXiv:1708.04552)
[26] Zhang H, Cisse M, Dauphin Y N and Lopez-Paz D 2017 mixup: beyond empirical risk minimization (arXiv:1710.09412)
[27] Cubuk E D, Zoph B, Mane D, Vasudevan V and Le Q V 2018 AutoAugment: learning augmentation policies from data (arXiv:1805.09501)
[28] Yin D, Lopes R G, Shlens J, Cubuk E D and Gilmer J 2019 A Fourier perspective on model robustness in computer vision (arXiv:1906.08989)
[29] DeVries T and Taylor G W 2017 Dataset augmentation in feature space (arXiv:1702.05538)
[30] Konno T and Iwazume M 2018 Icing on the cake: an easy and quick post-learning method you can try after deep learning (arXiv:1807.06540)
[31] Feng S Y, Ganapathy V, Wei J, Chandar S, Vosoughi S, Mitamura T and Hovy E 2021 A survey of data augmentation approaches for NLP (arXiv:2105.03075)
[32] Zhang X, Zhao J and LeCun Y 2015 Character-level convolutional networks for text classification Adv. in Neural Information Processing Systems vol 28 pp 649–57
Wang W Y and Yang D 2015 That’s so annoying!!: a lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using petpeeve tweets Proc. 2015 Conf. on Empirical Methods in Natural Language Processing pp 2557–63

Fadade M, Bisazza A and Monz C 2017 Data augmentation for low-resource neural machine translation (arXiv:1705.00440)

Senrich R, Haddow B and Birch A 2015 Improving neural machine translation models with monolingual data (arXiv:1511.06709)

Kafle K, Yousefhussein M and Kanan C 2017 Data augmentation for visual question answering Proc. 10th Int. Conf. on Natural Language Generation pp 198–202

Devlin J, Chang M-W, Lee K and Toutanova K 2018 BERT: pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805)

Chen T, Kornblith S, Norouzi M and Hinton G 2020 A simple framework for contrastive learning of visual representations Int. Conf. on Machine Learning (PMLR) pp 1597–607

Tian Y, Sun C, Poole B, Krishnan D, Schmid C and Isola P 2020 What makes for good views for contrastive learning? (arXiv:2005.10243)

McNaught A D, Lambard G and Gracheva E 2020 SMILES-X: autonomous molecular compounds characterization for small datasets without descriptors Mach. Learn.: Sci. Technol. 1 025004

Wu Z, Pan S, Chen F, Long G, Zhang C and Yu P 2020 A comprehensive survey on graph neural networks Mach. Learn.: Sci. Technol. 3 1503

Landrum G 2020 Rdkit: open-source cheminformatics Zenodo (https://doi.org/10.5281/zenodo.3732263) (Accessed 1 March 2020)

Bajusz D, Rácz A and Héberger K 2015 Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? Cheminformatics 1 12–12

Xiong Z, Wang D, Liu X, Zhong F, Wan X, Li X, Li Z, Luo X, Chen K and Jiang H et al 2019 Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism J. Med. Chem. 63 8749–60

Hu W, Liu B, Gomes J, Zitnik M, Liang P, Pande V and Leskovec J 2020 Strategies for pre-training graph neural networks Int. Conf. on Learning Representations

You Y, Chen T, Sui Y, Chen T, Wang Z and Shen Y 2020 Graph contrastive learning with augmentations Advances in Neural Information Processing Systems vol 33 pp 5812–23

Degen J, Wegscheid-Gerlach C, Zaliani A and Barey M 2008 On the art of compiling and using ‘drug-like’ chemical fragment spaces ChemMedChem 3 1503

Landrum G 2020 Rdkit: open-source cheminformatics Zenodo (https://doi.org/10.5281/zenodo.3732263) (Accessed 1 March 2020)

Bajusz D, Rácz A and Héberger K 2015 Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? Cheminformatics 1 12–12

McNaught A D and Atkins P 1997 Compendium of Chemical Terminology (Oxford: Blackwell Science)

Smith M B 2020 March’s Advanced Organic Chemistry: Reactions, Mechanisms and Structure (New York: Wiley)

Kim S, Noh J, Gu G G, Aspuru-Guzik A and Jang Y 2020 Generative adversarial networks for crystal structure prediction ACS Cent. Sci. 6 1412–20

Wu Z, Pan S, Chen F, Long G, Zhang C and Yu P 2020 A comprehensive survey on graph neural networks IEEE Trans. Neural Netw. Learn. Syst. 32 4–24

Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C and Sun M 2020 Graph neural networks: a review of methods and applications AI Open 1 57–81

Duvenaud D K, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A and Adams R P 2015 Convolutional networks on graphs for learning molecular fingerprints Advances in Neural Information Processing Systems vol 28

Kearnes S, McCloskey K, Berndl M, Pande V and Riley P 2016 Molecular graph convolutions: moving beyond fingerprints J. Comput. Aided Mol. Des. 30 595–608

Huang K, Xiao C, Glass L M, Zitnik M and Sun J 2020 Skipgnn: predicting molecular interactions with skip-graph networks Sci. Rep. 10 1–16

De Cao N and Kipf T 2018 MolGAN: an implicit generative model for small molecular graphs (arXiv:1805.11973)

You J, Liu B, Ying R, Pande V and Leskovec J 2018 Graph convolutional policy network for goal-directed molecular graph generation Proc. 32nd Int. Conf. on Neural Information Processing Systems pp 6412–22

Imre F, Bradley A R, van der Schaar M and Deane C M 2020 Deep generative models for 3D linker design J. Chem. Inf. Model. 60 1983–95

Hamilton W, Ying Z and Leskovec J 2017 Inductive representation learning on large graphs Advances in Neural Information Processing Systems

Li G, Muller M, Thabet A and Ghanem B 2019 DeepGCNs: Can GCNs go as deep as CNNs? Proc. IEEE/CVF Int. Conf. on Computer Vision pp 9267–76

Večerík P, Cucurull G, Casanova A, Romero A, Lio P and Bengio Y 2018 Graph attention networks Int. Conf. on Learning Representations

Gong L and Cheng Q 2019 Exploiting edge features for graph neural networks Proc. IEEE/CVF Int. Conf. on Computer Vision and Pattern Recognition pp 9211–9

Ying R, You J, Morris C, Ren X, Hamilton W L and Leskovec J 2018 Hierarchical graph representation learning with differentiable pooling Proc. 32nd Int. Conf. on Neural Information Processing Systems (NIPS’18) pp 4805–15

Gao H and Ji S 2019 Graph U-Nets Int. Conf. on Machine Learning (PMLR) pp 2083–92

Schmid J, Marques M R, Botti S and Marques M A 2019 Recent advances and applications of machine learning in solid-state materials science npj Comput. Mater. 5 1–36

Schlegel G R, Padilha A C, Acosta C M, Costa M and Fazzio A 2019 From DFT to machine learning: recent approaches to materials science—a review J. Phys. Mater. 2 032001

Chen C, Ye W, Zuo Y, Zheng C and Ong S P 2019 Graph networks as a universal machine learning framework for molecules and crystals Chem. Mater. 31 3564–72

St P C Phillips J C, Kemper T W, Wilson A N, Guan Y, Crowley M F, Nimlos M R and Larsen R E 2019 Message-passing neural networks for high-throughput polymer screening J. Chem. Phys. 150 234111

Park C W and Wolverton C 2020 Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery Phys. Rev. Mater. 4 063801
[77] Schütt K T, Kindermans P-J, Sauceda H E, Chmiela S, Tkatchenko A and Müller K-R 2017 SchNet: a continuous-filter convolutional neural network for modeling quantum interactions (arXiv:1706.08566)
[78] Li G, Xiong C, Thabet A and Ghanem B 2020 DeeperGCN: all you need to train deeper GCNs (arXiv:2006.07739)
[79] Chen D, Lin Y, Li W, Li P, Zhou J and Sun X 2020 Measuring and relieving the over-smoothing problem for graph neural networks from the topological view Proc. AAAI Conf. on Artificial Intelligence vol 34 pp 3438–45
[80] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 770–8
[81] Jain A et al 2013 Commentary: the materials project: a materials genome approach to accelerating materials innovation APL Mater. 1 011002
[82] Castelli I E, Olsen T, Datta S, Landis D D, Dahl S, Thygesen K S and Jacobsen K W 2012 Computational screening of perovskite metal oxides for optimal solar light capture Energy Environ. Sci. 5 5814–9
[83] Kim C, Huan T D, Krishnan S and Ramprasad R 2017 A hybrid organic–inorganic perovskite dataset Sci. Data 4 170057
[84] van der Maaten L and Hinton G 2008 Visualizing data using t-SNE J. Mach. Learn. Res. 9 2579–605