Supplementary Materials for

Examining graph neural networks for crystal structures: Limitations and opportunities for capturing periodicity

Sheng Gong et al.

Corresponding author: Sheng Gong, shengg@mit.edu; Jeffrey C. Grossman, jcg@mit.edu

Sci. Adv. 9, eadi3245 (2023)
DOI: 10.1126/sciadv.adi3245

This PDF file includes:

Supplementary Text
Figs. S1 to S6
Tables S1 to S6
1. Relationship between density, volume per atom and lattice

In Figure S1a, we show the correlation between lattice constants and density and volume per atom (“vpa”) of structures in the Materials Project database. We can see that, there is no obvious correlation between lattice parameters and density and vpa. As an example, in Figure S1b, we show the lattice constants of FCC- and HCP-closely packed structures. As we know, for elemental structures, FCC and HCP are the two most densely packed structures with the same density and packing fraction (0.74). However, FCC and HCP have different lattice constants and number of atoms in the Niggli primitive cell. This simple example illustrates that, two crystal structures can have very different lattices while share the same density and vpa if they have similar local coordination environments.

The phenomenon that GNN can learn density and vpa better than lattice constants might be attributed to the fact that, determination of lattice requires more global information than that of density and vpa. For example, to determine the vpa of FCC and HCP structures, information of the first coordination shell alone is enough, while to distinguish and determine the lattice of FCC and HCP structures, information of the second coordination shell is also required.
2. Limitation of average pooling on extensive properties.

Suggested by the limitation of average pooling on capturing primitive cell-level information, in this section, we show the limitation of average pooling on prediction of extensive properties. Extensive properties scale with number of atoms, such as total energy ($E_{\text{fin.}}$), (phonon) internal energy ($U$), (phonon) heat capacity ($C_v$) and magnetization ($M$). Despite their extensive nature, in many cases these extensive properties are
normalized to intensive versions for data storage and evaluation (16, 19, 28, 30, 35, 40, 87-89), such as energy per atom and magnetization per unit formula. Since intensive properties cannot be learned by sum pooling because of their size invariance (16), for convenience, many GNNs, such as CGCNN, ALIGNN and Matformer, only implement average pooling and learn the intensive versions of these extensive properties by average pooling in equation (4). Another way to learn extensive properties is illustrated in the bottom branch in Figure S2a, where extensive properties are first learned by machine learning models with sum pooling in equation (7), then normalized for evaluation.

Do the two approaches for learning extensive properties in Figure S2a have the same predictive power? Intuitively, they seem to be equivalent. However, as suggested in Figure 2c and 2h, we know that sum pooling is more powerful to distinguish graphs, and sum pooling can capture periodicity better. This insight suggests that sum pooling might be more powerful than average pooling for learning extensive properties. As shown in Figure S2b, we show the MAE/MAD scores of predictions of four extensive properties by CGCNN and ALIGNN with average pooling and sum pooling. We can see that, sum pooling outperforms average pooling for $U$, $C_v$ and $M$, and has similar performance for $E_{\text{fin}}$, which verifies our hypothesis that sum pooling might be more powerful than average pooling for learning extensive properties.

In order to further study the impact of these settings, and how hybridization with descriptors affects the regression with these different settings, we conduct further experiments about these settings on CGCNN, ALIGNN and their hybridized versions for $U$, $C_v$ and $M$. The results are listed in Table S1. Some notable trends are summarized below:

1) For predictions of $U$, $C_v$ and $M$ from original CGCNN and ALIGNN, compared with average pooling for property normalized per cell/formula, the improvement from sum pooling for not normalized property is more significant than average pooling for property normalized per atom. The reason might be the fact that, although the setting of per-atom normalization gets rid of the additional task of predicting number of atoms in the cell, average pooling is less capable to capture the periodicity of the structure than sum pooling, as suggested in Figure 2 in the main text.
2) When hybridized with descriptors, the difference between the three settings becomes smaller compared with that of the original GNNs. This is also reasonable, as when hybridized with descriptors, the information of number of atoms and periodicity is provided by descriptors, which partially replaces the role of per-atom normalization and sum pooling as discussed in 1).

3) Except de-ALIGNN for $M$ with sum pooling, in all other cases the descriptor-hybridized GNNs outperform the original GNNs. A possible reason for the irregular case is that, as in Figure 3c, when predicting $M$ the importance of descriptors is less than that of the automatically learned features from the GNNs. Therefore, for ALIGNN with sum pooling which already achieves a low predicting error, the positive impact of hybridizing with descriptors might be less significant than the negative impact of information redundancy as discussed below and in Figure S6 in the Supporting Information. Note that here we do not perform descriptors-selection, and we will investigate the impact of descriptors-selection on prediction performance in a future study.

Currently, many GNNs designed for prediction of materials properties, such as CGCNN, ALIGNN, and Matformer, only implement size-invariant average pooling function, with the exception of MEGNet where users can easily switch between sum pooling and average pooling. Based on the results shown in this section, we argue that GNN models designed for prediction of materials properties should provide options of pooling functions for different properties, such as average pooling for intensive properties and sum pooling for extensive properties.
Figure S2. Average pooling and sum pooling for extensive properties. a Schematic of two approaches to learn, predict and evaluate prediction performance of extensive properties. b MAE/MAD of prediction of four extensive properties by CGCNN and ALIGNN with average pooling and sum pooling.

Table S1. MAE of different regression settings for phonon internal energy ($U$), phonon heat capacity ($C_v$), and total magnetization ($M$) from CGCNN, de-CGCNN, ALIGNN, and de-ALIGNN. The value inside the parenthesis in each cell is the MAE/MAD ratio.

| Unit         | Pooling | CGCNN     | de-CGCNN  | ALIGNN | de-ALIGNN |
|--------------|---------|-----------|-----------|--------|-----------|
| $U$ KJ/mol-cell | Average | 0.71 ± 0.03 | **0.060 ± 0.005** | 0.53 ± 0.04 | 0.10 ± 0.01 |
| $U$ KJ/mol-atom | Average | 0.60 ± 0.03 | **0.059 ± 0.004** | 0.59 ± 0.04 | 0.12 ± 0.01 |
| $U$ KJ/mol-cell | Sum     | 0.25 ± 0.02 | **0.060 ± 0.005** | 0.27 ± 0.03 | 0.073 ± 0.006 |
| $C_v$ J/(mol-cell*K) | Average | 0.76 ± 0.04 | **0.058 ± 0.004** | 0.63 ± 0.04 | 0.089 ± 0.006 |
| $C_v$ J/(mol-atom*K) | Average | 0.60 ± 0.04 | **0.057 ± 0.004** | 0.61 ± 0.04 | 0.11 ± 0.01 |
| $C_v$ J/(mol-cell*K) | Sum     | 0.27 ± 0.02 | **0.051 ± 0.003** | 0.34 ± 0.03 | 0.067 ± 0.006 |
| $M$ μB/formula | Average | 0.41 ± 0.02 | 0.34 ± 0.01 | 0.34 ± 0.02 | **0.31 ± 0.01** |
| $M$ μB/atom   | Average | 0.34 ± 0.02 | **0.27 ± 0.02** | 0.30 ± 0.02 | 0.30 ± 0.01 |
| $M$ μB/cell  | Sum     | 0.30 ± 0.02 | 0.27 ± 0.01 | **0.25 ± 0.02** | 0.30 ± 0.02 |
3. Limitation of sum pooling on capturing periodicity.

As mentioned in the main text, even if only primitive cells are input to the GNNs, sum pooling might also fail to capture periodicity in some cases, as periodicity does not always scale with number of atoms in the primitive cells. For example, in Figure 2d we show the case of 1D double chains. Compared with 1D single chains in Figure 2b and 2c, 1D double chains can have similar periodicity but twice number of atoms. In Figure S3, we show the $R^2$ scores of predictions of $a$ of the datasets with 1D single short chains, 1D single and double short chains, 1D single long chains, and 1D single and double long chains, from CGCNN with average pooling and sum pooling, respectively. For the datasets with only single chains, sum pooling outperforms average pooling significantly, while for the datasets with single and double chains, sum pooling can only have very limited improvement over average pooling, which shows the co-existence of single and double chains makes sum pooling harder to determine periodicity than the case with only single chains.

![Figure S3](image)

**Figure S3.** $R^2$ scores of predictions of $a$ of the datasets with 1D single short chains, 1D single and double short chains, 1D single long chains, and 1D single and double long chains, from CGCNN with average pooling and sum pooling, respectively.
4. Niggli reduced primitive cell and its impact on learning performance of GNN.

As stated in the main text, we use the Niggli reduction to determine the unique Niggli-primitive cell for any given periodic structure\(^{(46)}\). In crystallography, we can use a set of six parameters to define a primitive cell: \((a, b, c, \alpha, \beta, \gamma)\), where the first three parameters are lengths of lattice vectors, and the later three parameters are angles between lattice vectors. In Niggli reduction, we require the lengths of lattice vectors to satisfy the following requirement:

\[
a + b + c = \text{minimum} \ldots \quad (S1)
\]

If the order of the three lengths are defined, as in the main text we define \(a\) the largest one and \(c\) the smallest one, then the set of \((a, b, c)\) is unique for any given structure\(^{(47)}\). In other words, the Niggli reduction requires that the primitive cell is composed by three shortest non-coplanar vectors. As shown in Figure S4a, in the example of 2-dimensional square lattice, the Niggli reduced cell is cell 1 (cell 2 and 3 are “primitive cells”, but not “Niggli reduced primitive cell”). The lengths of the shortest lattice vectors do have their specific physical meaning. For example, in terms of lattice vibration (acoustic phonon), the lengths of the shortest lattice vectors determine the normal modes of the vibration, such as the normal modes of lattice vibrations of 2-dimensional square lattice in equation S2\(^{(66)}\):

\[
\omega^2 = \frac{4K}{M} \sin^2 \left(\frac{k_x a}{2}\right) + \frac{4K}{M} \sin^2 \left(\frac{k_y a}{2}\right) \ldots \quad (S2),
\]

where \(\omega\) is the angular velocity of the normal modes, \(K\) is the spring constant, \(M\) is the mass of the primitive cell, and \(k_x\) and \(k_y\) are the quantum numbers of vibrations at the two dimensions, respectively. We can see that the normal modes are separated by \(\frac{a}{2}\), not \(\frac{\sqrt{2} a}{2}\) or lengths of other possible lattice vectors, which shows that the lattice lengths of the Niggli reduced primitive cell do have special physical meaning that lengths of other possible lattice vectors do not have. Therefore, our observation that GNN cannot capture lattice parameters of Niggli primitive cell is physically meaningful, which naturally leads to the fact that GNN cannot accurately predict properties related to lattice vibration as in the second half of our paper.

However, the set of \((\alpha, \beta, \gamma)\) might not be unique even if \((a, b, c)\) is unique. As mentioned in the main
text, both CGCNN and ALIGNN cannot learn the lattice angles well. In Figure S4b, we show the prediction performance of CGCNN, CGCNN with more convolutions, CGCNN with larger limits of number of neighbors and neighboring cut-off radius, CGCNN with sum pooling and ALIGNN for three lattice angles. We can see that both CGCNN and ALIGNN cannot learn the three lattice angles well, and the modifications that improve learning performance of lattice lengths as in the main text do not improve that of lattice angles. The artificial choice of lattice angles might cause the poor learning performance. For example, in Figure S4c, we plot the structure of simple hexagonal structure. We can see that the three lengths of lattice vectors are unique and reflect the intrinsic characteristics of the structure, such as the three minimal distances between the smallest repeating units in three dimensions. For \((\alpha, \beta, \gamma)\), we know that two of them are 90° (angles between the vertical lattice vector and the two in-plane lattice vectors), but there are actually two choices of the third one: 60° and 120°, and two choices of \((\alpha, \beta, \gamma)\): \((120°, 90°, 90°)\) and \((90°, 90°, 60°)\). Although in Niggli reduction, \((120°, 90°, 90°)\) is chosen as the lattice angles, this choice is artificial to the given structure without clear physical meaning, to the best of the authors’ knowledge. Therefore, machine learning algorithms might not be able to capture the artificially determined characteristics of crystal structures. Further studies are necessary to design more intrinsic description of relative orientation between the lattice vectors.
Figure S4. Primitive cell and lattice angles. a $R^2$ scores of prediction of lattice angles from CGCNN, ALIGNN and variants of CGCNN. Alpha denotes the largest lattice angle, and gamma the smallest one. b Illustrations of ambiguity of choice of lattice angles by the example of simple hexagonal primitive cell.

5. Receptive field of GNNs for inorganic crystal structures.

In order to estimate the length of the receptive field of GNNs in real 3D crystal structures, we estimate the average length of the receptive field by equations S3 and S4:

$$V = \frac{4}{3} \pi r^3 \approx n \ast vpa \quad \text{(S3)},$$

$$r = \begin{cases} 
\frac{3}{4\pi} \ast n \ast vpa, & \text{if } r < r_{\text{cut}} \\
 r_{\text{cut}}, & \text{if } r \geq r_{\text{cut}} 
\end{cases} \quad \text{(S4)},$$

where $V$ is the volume within the sphere, $n$ is the number of neighbors, $vpa$ is the volume per atom of the crystal structure, $r$ is the radius of the sphere (half of the length of the receptive field in one convolution step in GNNs), and $r_{\text{cut}}$ is the cut-off radius of the GNNs (8 Å for default CGCNN and ALIGNN). In Figure S5a, we show the distribution of $r$ with $n = 12$ (default maximum number of neighbors in CGCNN and ALIGNN),
from which we can see that the average $r$ is around 3.4 Å, therefore with three steps of convolutions the average length of receptive fields is around 20 Å.

As shown in Figure S5b, although most crystal structures in the Materials Project database have lattice around 10 Å, there are still some structures with long periodicity. For example, there are 6,000 structures with the longest lattice vector longer than 20 Å. Two examples of the long structures are provided in Figure S5c, which are $\text{Mo}_9\text{O}_{25}$ (mp-28777, $a = 28.4$ Å, energy above hull = 0.008 eV/atom) and $\text{Pr}_2\text{Au}_5\text{F}_{21}$ (mp-14715, $a = 26.3$ Å, energy above hull = 0 eV/atom). Therefore, our 1D toy dataset and analysis is still relevant to actual usages of GNNs as they represent thousands of materials where the receptive fields of typical GNNs are shorter than the periodicity of the crystal structures.

**Figure S5. Distribution of lattice constants.** a Distribution of the average radius to reach 12 neighbors of all structures in the Materials Project database. b Distribution of $a$ of all structures in the Materials Project database. c Two examples of structures with long periodicity. Color coding: red: O; purple: Mo; gold: Pr; orange: Au; grey: F.
6. Structural complexity

Structural complexity is defined as below:

\[ S = -N \sum_{i=1}^{k} \frac{m_i}{N} \log_2 \frac{m_i}{N} \ldots \text{(S5)}, \]

where \( N \) is the total number of atoms in the primitive cell, \( k \) is the number of symmetrically inequivalent sites, \( m_i \) is the number of sites classified in the \( i^{th} \) symmetrically inequivalent site. Structural complexity quantifies the complexity of sites distribution in a structure, as larger complexity, more different symmetrically inequivalent sites in the structure. Structural complexity per primitive cell is calculated as equation S5, and structural complexity per atom is calculated by equation S5 divided by the number of atoms in the primitive cell.

7. More discussions about \( U \) and \( C_v \)

Recently, Legrain et al.(67) reported machine learning of \( U \) by compositions of materials. The main difference between this work and Legrain et al.(67) is that, in Legrain et al.(67), only 292 materials are included in the dataset, while in this work about 1,500 materials are included in the datasets for \( U \) and \( C_v \). On the other hand, Mat2Spec(68) and E3NN(58) are proposed to predict phonon density of states, and consequently, heat capacity. Unfortunately, we cannot easily compare our predictions and predictions from the two models mentioned above, as the \( C_v \) in this work is based on full phonon density of states, while Mat2Spec(68) and E3NN(58) are designed to predict filtered and truncated phonon density of states with 51 frequencies up to 1000 cm\(^{-1}\). The reason for the success of E3NN for predicting \( C_v \) might be that E3NN employs equivariant representations(58) with high local expressive power, and the reason for the success of Mat2Spec for predicting \( C_v \) might be the explicit exploitation of correlations of density of states between frequencies in phonon density of states. Very recently, Gurunathan et al.(69) employ ALIGNN to predict phonon density of states, and find that \( C_v \) from predictions of phonon density of states is more accurate than \( C_v \) from direct prediction, which provides further insights into predictions of phonon related properties.
8. Impact of hybridization with descriptors on the learned representations

As in equation (8) in the main text, since the descriptors participate in the optimization of deep representation learning, the inclusion of descriptors would affect the optimization of the learned representations. In other words, consider the gradient propagation in the optimization of the representation learning:

$$\frac{\partial L}{\partial w_{pq}^R} = \frac{\partial z_{q}^{R+1}}{\partial w_{pq}^R} \times \frac{\partial L}{\partial z_{q}^{R+1}} \ldots \ (S6),$$

where $L$ is the loss function, $w_{pq}^R$ is the weight from the $p^{th}$ unit of the representation layer ($\frac{1}{N_a} \sum_{i=1}^{N_a} a_i^n \oplus$ descriptors) in equation (8) to the $q^{th}$ unit of the layer after the representation layer (first layer of FCN in equation (8)), and $z_{q}^{R+1}$ is the $q^{th}$ unit of the layer after the representation layer. Therefore, after inclusion of descriptors, $z_{q}^{R+1}$ changes, and consequently $\frac{\partial L}{\partial w_{pq}^R}$ changes even if the $p^{th}$ unit is from the part $\frac{1}{N_a} \sum_{i=1}^{N_a} a_i^n$, and all gradients before the representation layer change due to the chain rule of gradient propagation.

How does the change of gradient affect the learned representations? As an example, in Figure 3b, we show the feature importance spectrum of de-CGCNN for prediction of $\kappa$, from which we can see that the learned representations play the most important roles, and some descriptors contribute significantly such as mean and standard deviation of bond length, volume per atom and $b$. Except $b$, the other three important descriptors can be well learned by CGCNN as shown in Table 2 and Table 3. In order to understand this phenomenon, we investigate how well the learned representations from CGCNN correlate with descriptors before and after the hybridization with descriptors. In other words, we investigate the following correlation:

$$\text{Corr}(\frac{1}{N_a} \sum_{i=1}^{N_a} a_i^n, \text{descriptors}) \ldots \ (S7).$$

In Figure S6a, we show the correlation between the learned representations and each descriptor for learning $\kappa$, and we can see that the learned representations from de-CGCNN are less correlated with the descriptors than that from CGCNN. The weaker correlation after the inclusion of descriptors supports our hypothesis that
hybridization with descriptors pushes the optimization of learned representations away from the already known information in the input human-designed descriptors.

As a comparison, we construct machine learning models based on learned representations from CGCNN and human-designed descriptors as below:

\[
\text{Output} = \text{FCN}(\frac{1}{N_a} \sum_{i=1}^{N_a} q_i^n (\text{already learned from CGCNN}) \oplus \text{descriptors}) \quad \text{...... (S8).}
\]

The main difference between equation (S8) (named as “CGCNN+descriptors”) and equation (8) (de-CGCNN) is that, descriptors participate in the optimization of learned representations in de-CGCNN, while descriptors do not in CGCNN+descriptors. In Figure S6b, we show the MAE/MAD ratio of machine learning models based only on descriptors, CGCNN, CGCNN+descriptors, and de-CGCNN. We can see that, for most properties, de-CGCNN has lower error than CGCNN+descriptors, which supports the proposed mechanism that participation of descriptors in the optimization of representations improves the performance of deep representation learning. For \(\kappa\), \(U\), \(C_v\), and \(M\), we observe that CGCNN+descriptors outperforms CGCNN, and such improvement mainly comes from the missing information in descriptors as discussed in the main text. For most remaining properties, CGCNN+descriptors has similar performance with CGCNN, which suggests that the improvement of de-CGCNN for these properties might come from the participation of descriptors in the optimization of representations.

Despite the observed improvement, intuitively, hybridization with descriptors in equation (8) and addition of descriptors in equation (S8) would have a negative impact on prediction performance due to the introduction of redundant information from these two modifications\((77, 78)\). Such redundancy can be seen in Figure S6a, where learned representations from both CGCNN and de-CGCNN are correlated with human-designed descriptors in some degree. In Figure S6b, we observe that CGCNN+descriptors has higher error than the model based on only descriptors for \(U\) and \(C_v\). Since the only difference between the two models is the presence of learned representations in CGCNN+descriptors, the increase of error associated with CGCNN+descriptors supports the proposed mechanism that redundant information can harm the predictive power of machine learning models \((77, 78)\).
Figure S6. Influence of hybridization on learned representations. a $R^2$ scores of linear regressions between each descriptor and the learned representations trained on $\kappa$ from CGCNN and de-CGCNN. b MAE/MAD ratio of prediction of 13 materials properties from machine learning models based only on descriptors, CGCNN, machine learning models based on CGCNN-learned representations and descriptors, and de-CGCNN.
9. Pseudocode to generate the 1D chains

For each chain:

```python
pos = [] #initialize positions of atoms

for j in range(n): # number of atoms in the chain
    if j == 0:
        pos.append([3*random, 3*random, 3*random])
        # first atom, random coordinates in all three dimensions
        # 3 = 2*1.5 Å (1.5 Å to approximate C-C bond length)
        # random: random number between (0, 1)
    elif j%2 == 0:
        pos.append([pos[j-1][0] + 3*random, pos[j-1][1] + 3*random, pos[j-1][2] + 3*random])
        # even number of atom, random displacement from the previous
        # atom in all the three dimensions
    else:
        pos.append([pos[j-1][0] + 3*random, pos[j-1][1] - 3*random, pos[j-1][2] - 3*random])
        # odd number of atom, elongation in the first dimension, retraction in the other
        # two dimensions to keep the chain intact when creating lattice.

a = pos[-1][0]; b = 100; c = 100
# position of the last atom as the end of the cell, add vacuum for b and c

lattice = Lattice.from_parameters(a, b, c, 90, 90, 90)
```
10. Other supplementary tables

Table S2. Hyper-parameter search space for CGCNN, CGCNN with sum pooling, and de-CGCNN. Parameters not mentioned here are set to the default value as in the open source codes.

| Name                        | Space          |
|-----------------------------|----------------|
| atom feature length         | 32, 64         |
| hidden feature length       | 64, 128        |
| number of hidden layers     | 1, 2, 3        |
| learning rate               | 1e-3, 1e-2     |

Table S3. Hyper-parameter search space for ALIGNN, ALIGNN with sum pooling, and de-ALIGNN.

| Name                        | Space          |
|-----------------------------|----------------|
| edge input feature length   | 40, 80         |
| hidden feature length       | 64, 128, 256   |
| triplet input feature length| 20, 40         |
| learning rate               | 1e-3, 1e-2     |

Table S4. Hyper-parameter search space for MEGNet, state-MEGNet, and de-MEGNet.

| Name                        | Space          |
|-----------------------------|----------------|
| atom feature length         | 16, 32         |
| edge feature length         | 16, 32         |
| hidden feature length       | 32, 64         |
| number of hidden layers     | 1, 2, 3        |
| learning rate               | 1e-3, 1e-2     |

Table S5. Hyper-parameter search space for Matformer and de-Matformer.
| Name                              | Space               |
|----------------------------------|---------------------|
| edge input feature length        | 40, 80              |
| hidden feature length            | 64, 128, 256        |
| weight decay                     | 1e-6, 1e-5, 1e-4    |
| learning rate                    | 1e-3, 1e-4          |

Table S6. Hyper-parameter search space for E3NN and de-E3NN.

| Name                              | Space               |
|----------------------------------|---------------------|
| embedding length                 | 64, 32              |
| maximum order of spherical harmonics | 2, 3              |
| weight decay                     | 1e-6, 1e-5, 1e-4  |
| learning rate                    | 3e-3, 1e-3, 3e-4  |