Periodic Graph Transformers for Crystal Material Property Prediction

Keqiang Yan
Computer Science & Engineering
Texas A&M University
College Station, TX 77843
keqiangyan@tamu.edu

Yi Liu*
Computer Science
Florida State University
Tallahassee, FL 32306
liuy@cs.fsu.edu

Yuchao Lin
Computer Science & Engineering
Texas A&M University
College Station, TX 77843
kruskallin@tamu.edu

Shuiwang Ji*
Computer Science & Engineering
Texas A&M University
College Station, TX 77843
sji@tamu.edu

Abstract

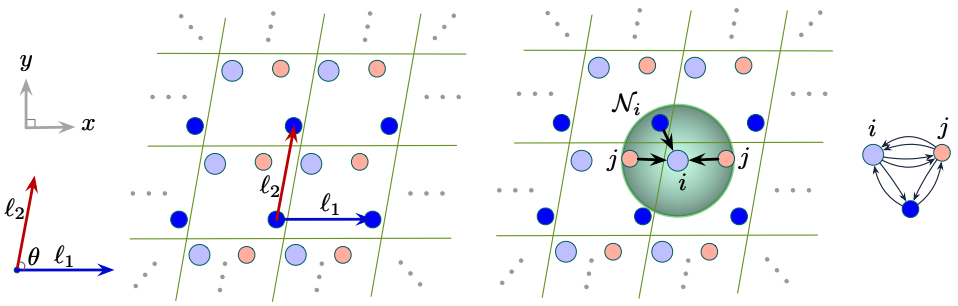
We consider representation learning on periodic graphs encoding crystal materials. Different from regular graphs, periodic graphs consist of a minimum unit cell repeating itself on a regular lattice in 3D space. How to effectively encode these periodic structures poses unique challenges not present in regular graph representation learning. In addition to being E(3) invariant, periodic graph representations need to be periodic invariant. That is, the learned representations should be invariant to shifts of cell boundaries as they are artificially imposed. Furthermore, the periodic repeating patterns need to be captured explicitly as lattices of different sizes and orientations may correspond to different materials. In this work, we propose a transformer architecture, known as Matformer, for periodic graph representation learning. Our Matformer is designed to be invariant to periodicity and can capture repeating patterns explicitly. In particular, Matformer encodes periodic patterns by efficient use of geometric distances between the same atoms in neighboring cells. Experimental results on multiple common benchmark datasets show that our Matformer outperforms baseline methods consistently. In addition, our results demonstrate the importance of periodic invariance and explicit repeating pattern encoding for crystal representation learning.

1 Introduction

Crystal material property prediction is important for the discovery of new materials with desirable properties [38, 35–37, 49, 51, 41, 4, 33, 6]. Different from molecules and proteins [15, 50, 42, 9, 48, 26, 44], which are commonly represented as graphs [45, 29, 11, 12, 25, 27, 34], crystals consist of a minimum unit cell repeating itself on a regular lattice in 3D space. Thus, crystals are naturally represented as periodic graphs. A key challenge of crystal material property prediction lies in how to effectively encode periodic structures that are not present in regular molecular graph representations [10, 20, 41, 23, 22, 31, 24, 39, 17, 13, 1]. E(3) invariance for molecular graphs requires the representation for a given molecule to be invariant to translation, rotation and reflection transformations in 3D space. Beyond that, periodic graphs of crystals require unique periodic

*Equal senior contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) (b) (c)

Figure 1: Illustrations of periodic patterns and the multi-edge graph construction method. Green lines are artificial boundaries to form one possible unit cell that repeats in infinite space for the given crystal. (a). An illustration of periodic patterns in 2D space. We use blue and red arrows to show how the blue atom repeats itself along ℓ_1 and ℓ_2. We show a general case in nature that ℓ_1 and ℓ_2 are not orthogonal. In this specific example, θ is less than $\frac{\pi}{2}$. (b) and (c). Illustrations of the multi-edge graph construction method and the constructed graph. The green circle shows atom i's neighborhood $\mathcal{N}_i$, and black arrows are edges from atom i's neighbors to itself. As atom j repeats twice within i's neighborhood in this example, the multi-edge graph construction method builds two edges from atom j to atom i. The crystals are 3D structures in practice, and we use illustrations in 2D for simplicity.

invariance. Periodic invariance has two facets; those are, the learned representations should be invariant to both the scaling up of the minimum repeatable unit cells and shifts of periodic boundaries. Although the former has been considered in Xie et al. [51, 52], the later is rarely identified explicitly and sometimes not considered by previous studies [51, 4, 46, 41, 6, 52, 33, 53, 1, 5]. Furthermore, the repeating patterns of periodic graphs should be captured explicitly. Given a fixed minimum unit cell structure, lattices of different sizes and orientations may correspond to different materials. Without such periodic patterns, the infinite structures of crystals may not be represented accurately. However, the explicit encoding of periodic patterns is not explored in previous studies [51, 4, 41, 6, 33, 1].

In this work, we propose to tackle periodic graph representation learning by incorporating both periodic invariance and periodic pattern encoding. We propose a periodic graph transformer, known as Matformer, that is periodic invariant and can capture periodic repeating patterns explicitly for crystal representation learning. Matformer achieves periodic invariance through two uniquely designed graph construction methods. It further encodes periodic patterns by efficient use of geometric distances between the same atoms in neighboring cells, thereby capturing the lattice size and orientation of a given crystal. We conduct experiments on two commonly used material benchmark datasets, including the Materials Project [16] and Jarvis [8]. Results show that Matformer outperforms baseline methods consistently on various tasks. In addition, our results demonstrate the importance of both periodic invariance and periodic pattern encoding for crystal representation learning.

2 Background

Crystal property prediction and crystal structures. Given a crystal represented as $(\mathbf{A}, \mathbf{P}, \mathbf{L})$, crystal property prediction aims to predict a target property value y, which is either real as $y \in \mathbb{R}$ or categorical as $y \in \{1, 2, \cdots, C\}$, for regression or classification task with C classes, respectively. Specifically, a crystal is represented as a unit cell with periodic patterns. A unit cell is a minimum repeatable structure for the given crystal, and it can be described by matrices $\mathbf{A}$ and $\mathbf{P}$. $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n]^T \in \mathbb{R}^{n \times d_a}$ is the atom feature matrix, where $\boldsymbol{a}_i \in \mathbb{R}^{d_a}$ is the d_a-dimensional feature vector for atom i in the unit cell. $\mathbf{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n]^T \in \mathbb{R}^{n \times 3}$ is the position matrix, where $\boldsymbol{p}_i \in \mathbb{R}^3$ contains the Cartesian coordinates for atom i in 3D space. To further encode periodic patterns, an additional lattice matrix $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3 \times 3}$ is used to describe how a unit cell repeats itself in three directions, including $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2$, and $\boldsymbol{\ell}_3$. We show a 2D case of periodic patterns for easy illustration in Fig. 1 (a). Note that crystals usually possess irregular shapes in practice. Hence, $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$, and $\boldsymbol{\ell}_3$ are not always orthogonal in 3D space. Formally, given a crystal representation

$(\mathbf{A}, \mathbf{P}, \mathbf{L})$, the infinite crystal structure can be represented as

$$\begin{aligned}\hat{\mathbf{P}} =& \{\hat{\boldsymbol{p}}_i | \hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}, i \in \mathbb{Z}, 1 \leq i \leq n\}, \\ \hat{\mathbf{A}} =& \{\hat{\boldsymbol{a}}_i | \hat{\boldsymbol{a}}_i = \boldsymbol{a}_i, i \in \mathbb{Z}, 1 \leq i \leq n\}.\end{aligned} \tag{1}$$

Here, $\hat{\mathbf{P}}$ contains all possible positions for each atom i, associated with the same $\boldsymbol{a}_i$ in $\hat{\mathbf{A}}$.

Multi-edge graph construction for crystals. The multi-edge graph construction proposed by Xie and Grossman [51] aims to capture atom interactions across cell boundaries, which are imposed artificially. In a regular molecular graph, a node corresponds to a single atom. In contrast, in a multi-edge graph, node i represents atom i and all its duplicates in the infinite 3D space. Apparently, node i contains the atom features vector $\boldsymbol{a}_i$ and all positions in the set $\{\hat{\boldsymbol{p}}_i | \hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$. Formally, the multi-edge graph construction method builds edges between nodes as follows. Given a prefixed radius $r \in \mathbb{R}$, if there exists any 3-tuple $(k_1^{'}, k_2^{'}, k_3^{'})$, where $k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{R}$, such that the Euclidean distance $d_{ji}^{'} \in \mathbb{R}$ satisfies $d_{ji}^{'} = ||\boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3 - \boldsymbol{p}_i||_2 \leq r$, an edge is built from j to i with the initial edge feature $d_{ji}^{'}$. An example of the edge construction is shown in Fig. 1 (b). Intuitively, if there exist m positions of node j within the radius of the center node i, this method builds m edges from node j to node i. By considering all possible positions of every node within a predefined radius in 3D space, the multi-edge graph construction method can in essence capture atom interactions across cell boundaries [40, 33, 4, 41, 6].

3 Periodic invariance and periodic pattern encoding for crystals

Different from molecular graphs, crystal graphs consist of a minimum unit cell repeating itself on a regular lattice in 3D space. When encoding such periodic structures, unique challenges lie in periodic invariance and periodic pattern encoding. In this section, we propose to formally define and analyze the importance of these two components.

3.1 Periodic invariance for crystals

Periodic invariance is proposed based on E(3) invariance, which is defined as below.

Definition 1 (Unit Cell E(3) Invariance)**.** *A function $f : (\mathbf{A}, \mathbf{P}, \mathbf{L}) \to \mathcal{X}$ is unit cell E(3) invariant such that for all $Q \in \mathbb{R}^{3\times3}, |Q| = \pm1$ and $b \in \mathbb{R}^3$, we have $f(\mathbf{A}, \mathbf{P}, \mathbf{L}) = f(\mathbf{A}, Q\mathbf{P} + b, Q\mathbf{L})$, where Q is rotation and reflection transformations, and b is translation transformations in 3D space.*

Intuitively, the structure of a cell remains the same when either applying rotations and reflections to position matrix $\mathbf{P}$ and lattice matrix $\mathbf{L}$ together, or applying translations to $\mathbf{P}$ only. Correspondingly, the output of the unit cell E(3) invariant function should remain the same.

In addition to unit cell E(3) invariance, periodic invariance is also shown necessary for generating valid crystal representations. Specifically, when the lattice matrix $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times3}$ is fixed for a crystal, we can still obtain different position matrices $\mathbf{P} \in \mathbb{R}^{n\times3}$ and different unit cell structures by shifting the period boundaries. As shown in Fig. 2 (a) and (b), the formed unit cell structures are different for the same crystal by shifting period boundaries. To this end, we further introduce periodic invariance, which shows that when the periodic boundaries are shifted or scaled up, the periodic invariant representation should remain the same. Formally, based on Sec. 2, we further define a function $\Phi : (\hat{\mathbf{A}}, \hat{\mathbf{P}}, \mathbf{L}, \mathrm{p}) \to (\mathbf{A}, \mathbf{P})$ simulating how to form different unit cells

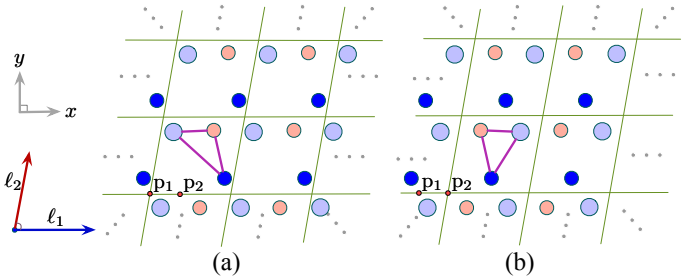
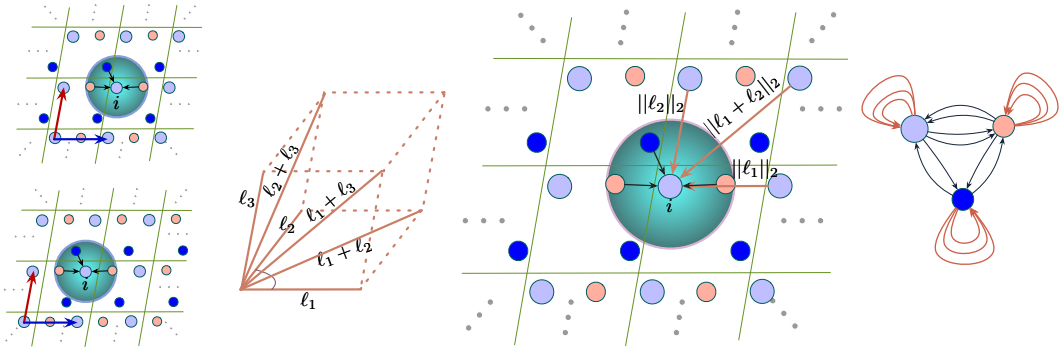


Figure 2: Illustration of periodic invariance. Purple lines are the edges between nodes inside a unit cell. Red points are the corner points of the unit cells. For example, p_1 and p_2 are for unit cells in (a) and (b), respectively. (a) and (b) show different unit cells describing the same crystal, caused by shifting the period boundaries along x axis from p_1 to p_2. By comparing (a) and (b), we show a graph construction method that breaks periodic invariance.



(a) (b) (c)

Figure 4: Illustrations of the used radius-based graph construction method and the proposed periodic pattern encoding for Matformer. Black arrows are the formed edges by radius-based graph construction and brown arrows are self-connecting edges. (a). Illustration that the used radius-based graph construction satisfies periodic invariance. (b). Illustration of periodic pattern encoding using self-connecting edges. On the left, we show the designed self-connecting edges to encode the lattice matrix $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times 3}$ in 3D space. We design six self-connecting edges $||\ell_1||_2$, $||\ell_2||_2$, $||\ell_3||_2$, $||\ell_1 + \ell_2||_2$, $||\ell_1 + \ell_3||_2$, $||\ell_2 + \ell_3||_2$. The geometric shape of $\mathbf{L}$ can be determined by these six edges. On the right, we show the added self-connecting edges in 2D space for easy illustration. (c). Illustration of the constructed graph with periodic pattern encoding for the above 2D case.

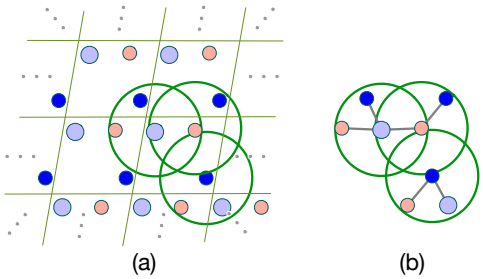
from a given infinite crystal structure. For an infinite crystal structure represented as $(\hat{\mathbf{A}}, \hat{\mathbf{P}})$, Φ uses a corner point p and shape matrix $\mathbf{L}$ to form a unit cell represented as $(\mathbf{A}, \mathbf{P})$. In addition, we use $\boldsymbol{\alpha} \in \mathbb{N}_+^3$ to indicate the scaling up of a repeating unit cell formed by periodic boundaries. Then the formal definition of periodic invariance is below.

Definition 2 (Periodic Invariance)**.** *A unit cell E(3) invariant function* $f : (\mathbf{A}, \mathbf{P}, \mathbf{L}) \to \mathcal{X}$ *is periodic invariant if* $f(\mathbf{A}, \mathbf{P}, \mathbf{L}) = f(\Phi(\hat{\mathbf{A}}, \hat{\mathbf{P}}, \boldsymbol{\alpha}\mathbf{L}, \mathrm{p}), \boldsymbol{\alpha}\mathbf{L})$ *holds for all* $\mathrm{p} \in \mathbb{R}^3$ *and* $\boldsymbol{\alpha} \in \mathbb{N}_+^3$.

Significance of periodic invariance. Breaking periodic invariance will result in different crystal graphs for the same crystal. In Fig. 2 (a) and (b), we show a graph construction method that breaks periodic invariance. This method is employed by a transformer-based model, known as Graphormer [53], which first uses radius to include all the atoms of interest in nearby cells and then builds a fully connected graph. A detailed illustration is shown in Fig. 6 in Appendix. A.1. Based on the formal definition of periodic invariance, in this study, we aim to integrate such an important component in our Matformer. By doing this, our model is able to construct a distinct crystal graph for a given crystal structure, resulting in a more informative and discriminative crystal learning scheme.

3.2 Periodic pattern encoding

As introduced in Sec. 2, $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times 3}$ containing periodic patterns is another key component to describe crystal structures. Essentially, periodic patterns show how the minimum repeatable structure $(\mathbf{A}, \mathbf{P})$ expands itself in infinite 3D space. Without such periodic pattern encoding in $\mathbf{L}$, crystal structures are treated as finite structures similar to molecules. As shown in Fig. 3, the widely used multi-edge graph construction method [51, 4, 41, 40, 6] only captures local interactions among atoms but ignores the important periodic patterns. However, such periodic repeating patterns need to be captured explicitly, as lattices of different sizes and orientations may correspond to different materials. Hence, To better represent the infinite structures of crystals, we argue that the periodic patterns $\mathbf{L} \in \mathbb{R}^{3\times 3}$ should be explicitly taken into consideration in crystal learning.



(a) (b)

Figure 3: Illustration that periodic patterns are not encoded in a multi-edge graph. Grey lines show the captured topology information by the multi-edge graph construction. We use three radius circles because there are three atoms in a unit cell. (a). Illustration of the multi-edge graph construction. (b). Illustration that the multi-edge graph method only captures local geometric information but ignores periodic patterns for the infinite structure.

4 The proposed Matformer

4.1 The proposed graph construction methods

In this section, we introduce the proposed graph construction methods for Matformer. Our methods effectively integrate periodic invariance and periodic pattern encoding.

Invariant crystal graph construction. We consider two crystal graph construction methods, including radius-based graph construction and fully connected graph construction. Both of them satisfy periodic invariance, and mathematical proofs can be found in Appendix. A.2. In Fig. 2, we show by example that treating atoms as single nodes breaks periodic invariance. Instead, our methods follow the fashion introduced in Sec. 2 to treat node i as atom i and all its repeated duplicates.

We use the multi-edge graph construction [51] introduced in Sec. 2 as an alternative crystal graph construction method. Note that although the multi-edge graph construction satisfies periodic invariance as in Fig. 4 (a), several existing works [51, 33] do not follow the settings exactly thus breaking periodic invariance. Within the given radius shown in Fig. 4 (a), these studies form the neighborhood of node i by selecting t nearest neighbors ranked by geometric distances. If there exist several different atoms with the same distance to node i, there is no deterministic way to select from them, as shown in Fig. 7 in Appendix. A.1. As a result, periodic invariance cannot be guaranteed.

Given the fully-connected fashion employed in Graphormer has achieved impressive performance on molecular learning, we further propose another graph construction method for Matformer, known as the fully connected graph construction. This method uses a different strategy to determine neighbors for each center node. Specifically, for node i and j, it builds edges for the entries corresponding to the t smallest distances in $\{d_{ij}|d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}\}$. It can be seen every pair i and j is connected in the constructed graph and there are t edges between them.

Overall, the used multi-edge graph construction and proposed fully-connected graph construction both satisfy the important periodic invariance. Particularly, they possess great flexibility to be used in future studies for crystal learning. In this study, we employ both methods as part of our proposed Matformer, and in main experiments, the radius-based method is used due to better empirical performance.

Periodic pattern encoding with self-connecting edges. In this study, we propose to encode the important $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times3}$ into crystal graphs by adding self-connecting edges. As mentioned in Sec. 3.2, periodic patterns describe sizes and orientations of lattices of a crystal structure, eventually determining the properties of this crystal. A natural step for encoding such repeating periodic patterns is to consider the relative positions between an atom and its nearby repeated duplicates. Formally, given an atom i with position $\boldsymbol{p}_i$ and $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times3}$, we need to encode the atom's three nearby duplicates with positions $\boldsymbol{p}_i + \boldsymbol{\ell_1}$, $\boldsymbol{p}_i + \boldsymbol{\ell_2}$, and $\boldsymbol{p}_i + \boldsymbol{\ell_3}$. It is widely known that a direction vector $\boldsymbol{\ell}_i$ is determined by both its length $||\boldsymbol{\ell}_i||_2$ and orientation. Essentially, $||\boldsymbol{\ell}_i||_2$ indicates the geometric distance between atom i and its corresponding duplicate. However, the computing of orientation information, such as angles, usually induces high complexity. Hence, it is not practical to encode such orientation information into transformer architectures. To this end, we propose to use geometric distances solely to implicitly consider the orientation information.

Specifically, we use additional distances to determine angles between any two direction vectors in $\mathbf{L}$. For example, the angle between $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ can be easily computed by $||\boldsymbol{\ell}_1||_2$, $||\boldsymbol{\ell}_2||_2$, and an additional distance $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2||_2$. Hence, based on these three distances, we can determine lengths of $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$, and the relative orientation between them. Extensively, we use six geometric distances, including $||\boldsymbol{\ell}_1||_2$, $||\boldsymbol{\ell}_2||_2$, $||\boldsymbol{\ell}_3||_2$, $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2||_2$, $||\boldsymbol{\ell}_2 + \boldsymbol{\ell}_3||_2$, and $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_3||_2$ in our study, as shown in Fig. 4 (b). By doing this, the length of each direction vector and the angle between any two direction vectors can all be determined. As a result, the shape formed by lattice matrix $\mathbf{L}$ is then fixed. Overall, we build the aforementioned six geometric distances as six self-connecting edges for node i. By doing this, our model is capable of encoding periodic patterns in $\mathbf{L}$ completely, resulting in a more accurate crystal representation learning scheme. Importantly, our approach also guarantees periodic invariance.

Overall, the graph construction for Matformer consists of two necessary stages, including invariant graph construction and periodic pattern encoding. For the first stage, we rigorously prove that the multi-edge graph construction satisfies periodic invariance in Appendix. A.2 and show that several previous works [51, 33] break periodic invariance using a different neighbor selection strategy. Additionally, we propose a fully-connected crystal graph construction method satisfying periodic

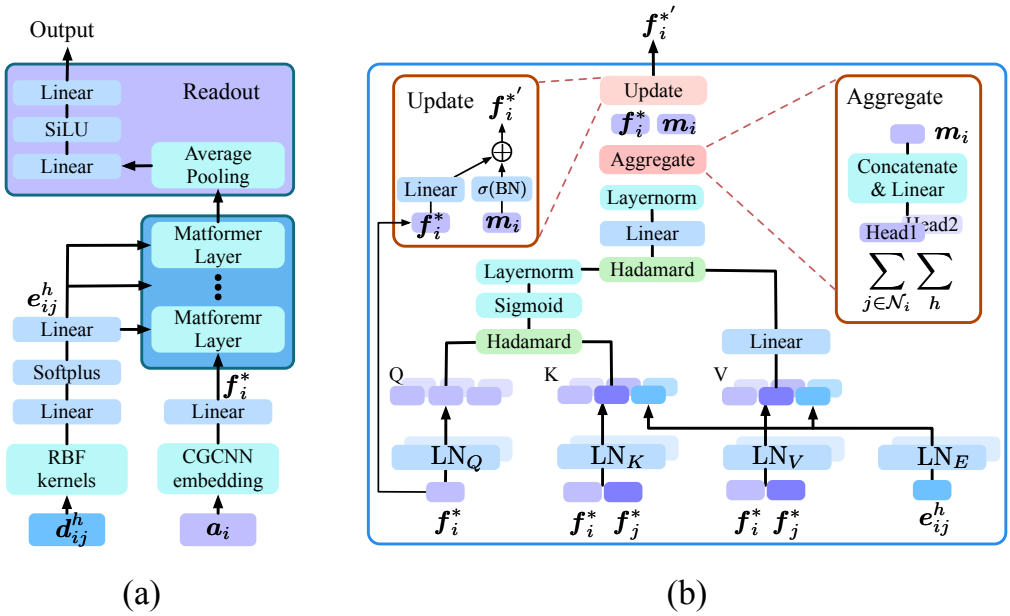


Figure 5: Illustration of detailed Architecture of Matformer. The overlapping graphics are used to denote two different attention heads. (a). Illustration of Matformer pipeline. (b). Illustration of the detailed Matformer layer in (a). We show the case with two attention heads for simplicity.

invariance, and the method could be used in future studies for crystal representation learning. For the second stage, we naturally encode periodic patterns into constructed graphs by adding self-connecting edges without breaking periodic invariance. A constructed graph in 2D case is shown in Fig. 4 (c).

4.2 Message passing scheme

Building on constructed graphs introduced in Sec. 4.1, we propose our message passing scheme for Matformer. Formally, we denote a constructed crystal graph as $G = (\mathbf{A}, \mathbf{E})$. Here, each $\boldsymbol{a}_i \in \mathbf{A}$ is the d_a-dimensional feature vector for atom i, as introduced in Sec. 2. Particularly, $\boldsymbol{e}_{ij}^h \in \mathbf{E}$ is d_e-dimensional feature vector for the h-th edge between nodes i and j. We follow the regular attention mechanism that computes query, key, and value [30, 28]. Our proposed message passing scheme is composed of three steps; those are, edge-wise attention coefficients computing, edge-wise value message computing, and node updating. Formally, we let $\boldsymbol{f}_i^{\star\ell}$ denote the input feature vector of node i for the ℓ-th layer of Matformer. The message passing scheme of the ℓ-th layer is described as below.

In the first step, $\boldsymbol{q}_{ij}^h$, $\boldsymbol{k}_{ij}^h$ and $\boldsymbol{\alpha}_{ij}^h$ for the h-th edge between i and j are computed as

$$\begin{aligned} \boldsymbol{q}_i =& \mathrm{LN}_Q(\boldsymbol{f}_i^{\star\ell}),\ \boldsymbol{k}_i = \mathrm{LN}_K(\boldsymbol{f}_i^{\star\ell}),\ \boldsymbol{k}_j = \mathrm{LN}_K(\boldsymbol{f}_j^{\star\ell}),\ \boldsymbol{e}_{ij}^{h'} = \mathrm{LN}_E(\boldsymbol{e}_{ij}^h), \\ \boldsymbol{q}_{ij}^h =& (\boldsymbol{q}_i|\boldsymbol{q}_i|\boldsymbol{q}_i),\ \ \boldsymbol{k}_{ij}^h = (\boldsymbol{k}_i|\boldsymbol{k}_j|\boldsymbol{e}_{ij}^{h'}),\ \ \boldsymbol{\alpha}_{ij}^h = \frac{\boldsymbol{q}_{ij}^h \circ \boldsymbol{k}_{ij}^h}{\sqrt{d_{\boldsymbol{k}_{ij}^h}}}, \end{aligned} \tag{2}$$

where LN_Q, LN_K, and LN_E denote the linear transformations to compute query, key, and edge embedding in ℓ-th layer, respectively. $\boldsymbol{e}_{ij}^{h'}$ is the intermediate output for $\boldsymbol{e}_{ij}^h$. We use $\circ$ and $|$ to denote Hadamard Product and concatenation. Note that $\boldsymbol{q}_{ij}^h$ is the concatenation of three $\boldsymbol{q}_i$ vectors to match the dimension of $\boldsymbol{k}_{ij}^h$. By doing this, when computing $\boldsymbol{\alpha}_{ij}^h$, $\boldsymbol{q}_i$ attends each of $\boldsymbol{k}_i$, $\boldsymbol{k}_j$, and $\boldsymbol{e}_{ij}^{h'}$ for integrating more information in attention.

Particularly, we omit the softmax to enhance the model's capability to distinguish nodes with different degrees, and to make the whole network more efficient.

After obtaining coefficients $\boldsymbol{\alpha}_{ij}^h$, in the second step, we compute $\boldsymbol{m}_{ij}^h$ that is the message of $\boldsymbol{e}_{ij}^h$ as

$$\boldsymbol{v}_i = \mathrm{LN}_V(\boldsymbol{f}_i^{\star\ell}),\ \boldsymbol{v}_j = \mathrm{LN}_V(\boldsymbol{f}_j^{\star\ell}), \boldsymbol{m}_{ij}^h = \mathrm{sigmoid}(\mathrm{LNorm}(\boldsymbol{\alpha}_{ij}^h)) \circ \mathrm{LN}_{\mathrm{update}}(\boldsymbol{v}_i|\boldsymbol{v}_j|\boldsymbol{e}_{ij}^{h'}), \tag{3}$$

where LN_V and $\text{LN}_{\text{update}}$ are the linear transformations to compute value and the updated message, and LNorm denotes the layer normalization operation.

Finally, in the third step, we compute node i's feature vector $\boldsymbol{f}_i^\ell$. Specifically, we first obtain message $\boldsymbol{m}_i$ by aggregating information from node i's neighborhood over multiple edges, then achieve $\boldsymbol{f}_i^\ell$ as

$$\boldsymbol{m}_i = \sum_{j \in \mathcal{N}_i} \sum_h \text{LNorm}(\text{LN}_{\text{msg}}(\boldsymbol{m}_{ij}^h)), \ \boldsymbol{f}_i^\ell = \text{LN}_{\text{fea}}(\boldsymbol{f}_i^{\star\ell}) + \sigma(\text{BN}(\boldsymbol{m}_i)), \tag{4}$$

where σ is the used activation function, and BN indicates batch normalization. In addition, LN_{msg} and LN_{fea} are linear transformations to update the messages on edges and the old atom features.

Graphormer represents an effective transformer variant for molecular graph learning. The differences between Graphormer and the proposed Matformer lie in both graph construction and message passing scheme. Firstly, Graphormer treats every atom as a single node and breaks periodic invariance, as mentioned in Sec. 3.1. For the message passing, Graphormer uses the node-wise attention and encodes pairwise distances as attention bias. It cannot work properly on multi-edge graphs for crystals. While Matformer is specifically designed for multi-edge crystal graphs by performing edge-wise attention and encoding geometric information into edge-wise messages, as described above. The detailed architecture of our Matformer is shown in Fig. 5.

5 Related work

Crystal property prediction. Several existing methods [46, 18, 19, 14] model crystals as chemical formulas and employ sequence models to process them. Other studies [51, 40, 33, 4, 41, 6] consider 3D structures and formulate crystals as 3D graphs, then apply GNNs to learn from crystal graphs. As crystals are essentially periodically repeated structures, the graph construction needs to consider periodic invariance and periodic pattern encoding. There are limited efforts to identify these two unique components. As an early work, CGCNN [51] proposes to capture atom interactions across artificial cell boundaries by using multi-edge graphs described in Sec. 2. The multi-edge graph satisfies periodic invariance as described in Sec. 3.1, but fails to consider the important periodic patterns, as described in Sec. 3.2. The multi-edge graph construction method is widely used in the following studies [40, 33, 4, 41, 6, 1]. Based on the constructed crystal graphs, many GNN variants have been proposed for effective crystal representation learning [51, 40, 33, 4, 41, 6, 1]. Specifically, Nequip [1] considers E(3) equivariance for materials, and satisfies periodic invariance using multi-edge graphs, but fails to capture periodic repeating patterns. We also notice a recent work [47] for periodic graph generation, which considers periodic graphs as finite graphs and breaks periodic invariance. Recently, ALIGNN [6] achieves the best performance on two major material datasets. It uses angle information in the message passing to generate more informative and discriminative representations. However, the use of angles introduces excessive time complexity.

Geometric GNNs and graph transformer. Many efforts have been made to incorporate 3D geometric information in molecular learning. Exemplary studies include SchNet [41], DimeNet [23, 22], SphereNet [31], GemNet [24], etc. However, these methods are designed for molecules without periodic patterns. Recently, graph transformers [53**?**] using geometric information, e.g., Graphormer [53], have shown great potential on real-world graph data. However, Graphormer considers neither periodic invariance nor periodic pattern encoding.

Differences with our method. To the best of our knowledge, periodic invariance and periodic pattern encoding described in Sec. 3 are rarely identified and explored in existing works for crystal property prediction. CGCNN [51] breaks periodic invariance on some corner cases because it uses twelve nearest neighbors determined only by distances as described in Sec. 4.1. In addition, previous methods including CGCNN [51], SchNet [41], MEGNET [4], CYATT [40], GATGNN [33], NEQUIP [1] and ALIGNN [6], all fail to consider the important periodic pattern encoding as introduced in Sec. 3.2. Especially, following GAT [45], GATGNN [33] employs a very limited kind of attention mechanism that is not conditioned on query, as explained in GATv2 [2]. As a result, the model capacity is reduced compared with the self attention mechanism employed in Matformer. In addition, the usage of softmax limits the capability of GATGNN of distinguishing nodes with different degrees, as mentioned in Sec. 4.2. For Graphormer [53], although it achieved remarkable success on the Open Catalyst Challenge [3], the employed graph construction method breaks periodic invariance when

Table 1: Comparison in terms of test MAE on The Materials Project dataset. To make the comparison clear and fair, We show results from retrained models using exactly the same training, validation, and test sets. Results from original papers are shown in Appendix A.5. The best results are shown in **bold** and the second best results are shown with underlines.

	Formation Energy	Band Gap	Bulk Moduli	Shear Moduli
Method	eV/atom	eV	log(GPa)	log(GPa)
CGCNN [51]	0.031	0.292	0.047	0.077
SchNet [41]	0.033	0.345	0.066	0.099
MEGNET [4]	0.030	0.307	0.060	0.099
GATGNN [33]	0.033	0.280	0.045	0.075
ALIGNN [6]	0.022	0.218	0.051	0.078
Matformer	**0.021**	**0.211**	**0.043**	**0.073**

applied to crystals. Compared with Graphormer, Matformer is specifically designed for crystals considering both periodic invariance and periodic patterns.

6 Experimental studies

6.1 Experimental setup

We conduct experiments on two material benchmark datasets, including The Materials Project [16] and JARVIS [8]. The detailed descriptions for The Materials Project and JARVIS datasets are shown in Appendix. A.3 Baseline methods include CFID [7], CGCNN [51], SchNet [41], MEGNET [4], GATGNN [33], and ALIGNN [6]. Unless otherwise specified, for all the baseline methods, we report the results taken from the referred papers or provided by original authors. All Matformer models are trained using the Adam optimizer [21] with weight decay [32] and one cycle learning rate scheduler [43]. We only slightly adjust learning rates from 0.001 and training epochs from 500 for different tasks. Detailed Matformer configurations for different tasks are provided in Appendix. A.4.

6.2 Experimental results

The Materials Project. We first use The Materials Project-2018.6.1 dataset [4], which contains 69239 crystals, to evaluate Matformer. We notice that previous works [51, 41, 4, 33, 6] compare with each other either using datasets of different sizes, or using datasets with the same size but splitting the datasets with different random seeds. To make the comparison clear and fair, we retrain all corresponding models using exactly the same training, validation and test sets across all methods and report the results in Table. 1. To avoid confusion, we still put original results from referred papers in Table. 1 inside parentheses. For retrained baseline models, we provide the detailed configurations in Appendix. A.5. The used metric is test MAE following previous studies [51, 41, 4, 33, 6].

It can be seen from Table. 1 that Matformer achieves the best performance on all tasks consistently by significant margins. Specifically, it reduces the formation energy by 4.5% of the second best model, which is a significant margin. Furthermore, for Bulk Moduli and Shear Moduli tasks with only 4664 training samples, Matformer achieves the best performances, indicating Matformer's adaptive ability to tasks of small training scales.

JARVIS dataset. The quantitative results for Jarvis are shown in Table. 2. Matformer outperforms the baseline methods significantly on all of these five tasks. Compared with ALIGNN, Matformer has stronger discriminative ability due to explicit encoding of periodic patterns. Specifically, Matformer reduces Jarvis Ehull by 0.012, which is 15.8% of ALIGNN. Furthermore, Matformer achieves the best performances for Bulk Moduli and Shear Moduli in the Mateirals Project with 4664 training samples, and Bandgap(MBJ) in JARVIS with 14537 training samples, indicating its adaptive ability to tasks of various data scales. Overall, the superior performances show the effectiveness of periodic pattern encoding in our Matformer message passing. In addition, compared with ALIGNN, our Matformer is more efficient. We evaluate the efficiency of Matformer by comparing with ALIGNN using JARVIS formation energy dataset. The mean time of ten runs for training and inference using best model configurations of ALIGNN and Matformer are reported. We also report the total number of parameters of each model. In Table. 3, we show that Matformer is three times faster than ALIGNN

Table 2: Comparison between Matformer and other baselines in terms of test MAE on JARVIS dataset. The best results are shown in **bold** and the second best results are shown with underlines.

	Formation Energy	Bandgap(OPT)	Total Energy	Ehull	Bandgap(MBJ)
Method	eV/atom	eV	eV/atom	eV	eV
CFID [7]	0.14	0.30	0.24	0.22	0.53
CGCNN [51]	0.063	0.20	0.078	0.17	0.41
SchNet [41]	0.045	0.19	0.047	0.14	0.43
MEGNET [4]	0.047	0.145	0.058	0.084	0.34
GATGNN [33]	0.047	0.17	0.056	0.12	0.51
ALIGNN [6]	0.0331	0.142	0.037	0.076	0.31
Matformer	**0.0325**	**0.137**	**0.035**	**0.064**	**0.30**

Table 3: Efficiency comparison with ALIGNN on Jarvis Formation Energy task. We show the training time per epoch, total training time, inference time for the whole test set, and total number of parameters.

Models	Time/epoch	Total	Inference	Model Para.
ALIGNN	327 s	27.3 h	156 s	15.4 MB
Matformer	64 s	8.9 h	59 s	11.0 MB

in total training time and near three times faster in inference time, for the whole test set. Matformer is also much lighter than ALIGNN in terms of model size.

Energy within Threshold. Following OC20 [3], we use energy within threshold (EwT), which measures the percentage of estimated energies that are likely to be practically useful when the absolute error is within a certain threshold, to evaluate Matformer's capability for periodic graph learning. This metric is new, but is well recognized by the community as it is useful in practice. Due to significant performance gaps between ALIGNN and other baseline methods on formation energy and total energy for these two datasets in terms of mean absolute error (MAE), we only compare Matformer with ALIGNN. Table. 4 shows that Matformer outperforms ALIGNN consistently for all three energy prediction tasks. Interestingly, the performance gains of our Matformer beyond ALIGNN in terms of EwT mainly come from more accurate energy predictions within absolute error of 0.01. Compared with JARVIS, the Materials Project has 15422 more traning samples. As a result, the percentage of predicted energies obtained by Matformer within 0.01 increases by 14.69%, which is much better than ALIGNN, revealing the huge potential of Matformer when larger crystal dataset is available.

6.3 Ablation studies

In this section, we demonstrate the importance of periodic invariance and explicit repeating pattern encoding for crystal representation learning by ablation studies. We also evaluate the building blocks particularly designed for our Matformer. Specifically, we conduct experiments on JARVIS formation energy, and the test MAE is used as the quantitative evaluation metric. We also provide ablation studies on the use of sigmoid and layernorm instead of softmax in Matformer layer in Appendix A.6.

Periodic invariant graph construction. We demonstrate the importance of periodic invariant graph construction by comparing radius multi-edge graph, denoted as Radius, with the graph construction method proposed by Graphormer, denoted as OCgraph, on the exactly same Matformer architecture. Note that the constructed crystal graphs by OCgraph are super large, containing more than n^2 edges,

Table 4: Comparison between Matformer and ALIGNN in terms of EwT on JARVIS Formation Energy, JARVIS Total Energy and The Materials Project Formation Energy. We use EwT (0.02) to mark the threshold of 0.02 and EwT (0.01) to mark the threshold of 0.01. The best results are in **bold**.

	Formation MP		Formation JARVIS		Total JARVIS	
Method	EwT (0.01)	EwT (0.02)	EwT (0.01)	EwT (0.02)	EwT (0.01)	EwT (0.02)
ALIGNN	49.94%	71.10%	39.59%	59.64%	35.09%	55.20%
Matformer	**55.86%**	**75.02%**	**41.17%**	**60.25%**	**36.84%**	**57.36%**

Table 5: Ablation studies on periodic invariance and periodic pattern encoding. We use OCgraph to denote graph construction method proposed by Graphormer. PI denotes periodic invariance and PE denotes periodic encoding.

Graph	PI	PE	layer	head	batch	Test MAE
OCgraph	×	×	3	1	32	0.0530
Radius w/o PE	✓	×	3	1	32	0.0348
Radius w/o PE	✓	×	5	4	64	0.0337
T-fully w PE	✓	✓	5	4	64	0.0402
Radius w PE	✓	✓	5	4	64	**0.0325**

where n is the atom number in a cell. We adjust the Matformer configurations to train these large graphs on a single RTX A6000 GPU. It can be seen from Table. 5 that when using OCgraph to our Matformer, the test MAE drops dramatically of 53% because of breaking periodic invariance, compared with radius-based multi-edge graphs in Matformer. We also compare two periodic invariant graph construction methods described in Sec. 4.1. We denote fully connected graph with t smallest pairwise distances as T-fully, and use $t = 3$. The result shows that Radius is better than T-fully.

Encoding of repeating patterns. We denote periodic pattern encoding as PE. In Table. 5, we show that omitting the periodic pattern encoding results in a significant drop of test MAE from 0.0325 to 0.0337, revealing the importance of periodic patterns for crystal representation learning.

Complexity of introducing angular information. Dropping angular information largely improves running efficiency of Matformer compared with ALIGNN. We show that for the original crystal graph with n nodes and $6n$ edges, the corresponding line graph with angles will have $6n$ nodes and $66n$ edges, leading to high computational cost. The detailed complexity analysis of adding angular information are provided in Appendix. A.7. Additionally, we provide the running time and performance analysis of Matformer with angle information in Table. 6. We use two Matformer layers to process extra angular information, and use Radial Basis Function kernels [6] and Spherical Bessel Functions with Spherical Harmonics [23, 31, 24] to encode angles, denoted as Matformer + Angle RBF and Matformer + Angle SBF. Table. 6 shows that introducing angular information will increase both the training time per epoch and in total by around 3 times, without much performance gain. This may due to the periodic invariant graph construction and periodic patterns encoding in Matformer already capture sufficient information to identify crystal structures.

Table 6: Ablation studies on angular information. We show the training time per epoch, total training time, and test MAE.

Models	MAE	Time/epoch	Total
Matformer	.0325	64 s	8.9 h
Matformer + Angle SBF	.0332	173 s	23.8 h
Matformer + Angle RBF	.0325	165 s	22.9 h

7 Conclusion and limitation

In this work, we first propose to formally define periodic invariance and periodic pattern encoding for periodic graph learning. We then propose Matformer for periodic graph representation learning, which is invariant to periodicity and can capture repeating patterns explicitly. Experimental results on common benchmark datasets show that our Matformer outperforms baseline methods consistently. In addition, our results demonstrate the importance of periodic invariance and explicit periodic pattern encoding for crystal representation learning. One potential direction beyond this work is to include angular information properly to satisfy both periodic invariance and to encode periodic patterns with relatively low time complexity, and this is one limitation of our work. Besides, negative societal impacts of material discovery may apply to our work.

Acknowledgments and Disclosure of Funding

We thank Tian Xie for answering our questions about CGCNN. This work was supported in part by National Science Foundation grant IIS-2006861.

References

[1] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. Nature communications, 13(1): 1–11, 2022.

[2] Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks? In International Conference on Learning Representations, 2021.

[3] Lowik Chanussot*, Abhishek Das*, Siddharth Goyal*, Thibaut Lavril*, Muhammed Shuaibi*, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open Catalyst 2020 (OC20) Dataset and Community Challenges. ACS Catalysis, 2021. doi: 10.1021/acscatal.0c04525.

[4] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. Chemistry of Materials, 31(9):3564–3572, 2019.

[5] Zhantao Chen, Nina Andrejevic, Tess Smidt, Zhiwei Ding, Qian Xu, Yen-Ting Chi, Quynh T Nguyen, Ahmet Alatas, Jing Kong, and Mingda Li. Direct Prediction of Phonon Density of States With Euclidean Neural Networks. Adv. Sci, 8:2004214, 2021.

[6] Kamal Choudhary and Brian DeCost. Atomistic Line Graph Neural Network for improved materials property predictions. npj Computational Materials, 7(1):1–8, 2021.

[7] Kamal Choudhary, Brian DeCost, and Francesca Tavazza. Machine learning with force-field-inspired descriptors for materials: Fast screening and mapping energy landscape. Physical review materials, 2(8):083801, 2018.

[8] Kamal Choudhary, Kevin F Garrity, Andrew CE Reid, Brian DeCost, Adam J Biacchi, Angela R Hight Walker, Zachary Trautt, Jason Hattrick-Simpers, A Gilad Kusne, Andrea Centrone, et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. npj Computational Materials, 6(1):1–13, 2020.

[9] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein Interface Prediction using Graph Convolutional Networks. Advances in Neural Information Processing Systems, 30, 2017.

[10] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. SE (3)-transformers: 3d roto-translation equivariant attention networks. Advances in Neural Information Processing Systems, 33:1970–1981, 2020.

[11] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In International Conference on Machine Learning, pages 2083–2092. PMLR, 2019.

[12] Hongyang Gao, Yi Liu, and Shuiwang Ji. Topology-Aware Graph Pooling Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(12):4512–4518, 2021.

[13] Mario Geiger and Tess Smidt. e3nn: Euclidean Neural Networks. arXiv preprint arXiv:2207.09453, 2022.

[14] Rhys EA Goodall and Alpha A Lee. Predicting materials properties without crystal structure: Deep representation learning from stoichiometry. Nature communications, 11(1):1–9, 2020.

[15] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In Proceedings. 2005 IEEE international joint conference on neural networks, volume 2, pages 729–734, 2005.

[16] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. APL materials, 1(1):011002, 2013.

[17] Priyank Jaini, Lars Holdijk, and Max Welling. Learning Equivariant Energy Based Models with Equivariant Stein Variational Gradient Descent. Advances in Neural Information Processing Systems, 34, 2021.

[18] Dipendra Jha, Logan Ward, Arindam Paul, Wei-keng Liao, Alok Choudhary, Chris Wolverton, and Ankit Agrawal. Elemnet: Deep learning the chemistry of materials from only elemental composition. Scientific reports, 8(1):1–13, 2018.

[19] Dipendra Jha, Logan Ward, Zijiang Yang, Christopher Wolverton, Ian Foster, Wei-keng Liao, Alok Choudhary, and Ankit Agrawal. IRNet: A general purpose deep residual regression framework for materials discovery. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2385–2393, 2019.

[20] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. Advances in Neural Information Processing Systems, 32, 2019.

[21] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, 2015.

[22] Johannes Klicpera, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. arXiv preprint arXiv:2011.14115, 2020.

[23] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In International Conference on Learning Representations, 2020.

[24] Johannes Klicpera, Florian Becker, and Stephan Günnemann. GemNet: Universal Directional Graph Neural Networks for Molecules. In Advances in Neural Information Processing Systems, 2021.

[25] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards Deeper Graph Neural Networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2020.

[26] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, et al. DIG: a turnkey library for diving into graph deep learning research. Journal of Machine Learning Research, 22(240):1–9, 2021.

[27] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. GraphEBM: Molecular graph generation with energy-based models. arXiv preprint arXiv:2102.00546, 2021.

[28] Yi Liu and Shuiwang Ji. CleftNet: Augmented Deep Learning for Synaptic Cleft Detection from Brain Electron Microscopy. IEEE Transactions on Medical Imaging, 40(12):3507–3518, 2021.

[29] Yi Liu, Hao Yuan, Lei Cai, and Shuiwang Ji. Deep Learning of High-order Interactions for Protein Interface Prediction. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pages 679–687, 2020.

[30] Yi Liu, Hao Yuan, Zhengyang Wang, and Shuiwang Ji. Global Pixel Transformers for Virtual Staining of Microscopy Images. IEEE Transactions on Medical Imaging, 39(6):2256–2266, 2020.

[31] Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In International Conference on Learning Representations, 2021.

[32] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In International Conference on Learning Representations, 2018.

[33] Steph-Yves Louis, Yong Zhao, Alireza Nasiri, Xiran Wang, Yuqi Song, Fei Liu, and Jianjun Hu. Graph convolutional neural networks with global attention for improved materials property prediction. Physical Chemistry Chemical Physics, 22(32):18141–18148, 2020.

[34] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A Discrete Flow Model for Molecular Graph Generation. In International Conference on Machine Learning, pages 7192–7203. PMLR, 2021.

[35] Bryce Meredig, Ankit Agrawal, Scott Kirklin, James E Saal, Jeff W Doak, Alan Thompson, Kunpeng Zhang, Alok Choudhary, and Christopher Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. Physical Review B, 89(9):094104, 2014.

[36] Anton O Oliynyk, Erin Antono, Taylor D Sparks, Leila Ghadbeigi, Michael W Gaultois, Bryce Meredig, and Arthur Mar. High-throughput machine-learning-driven synthesis of full-heusler compounds. Chemistry of Materials, 28(20):7324–7331, 2016.

[37] Paul Raccuglia, Katherine C Elbert, Philip DF Adler, Casey Falk, Malia B Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A Friedler, Joshua Schrier, and Alexander J Norquist. Machine-learning-assisted materials discovery using failed experiments. Nature, 533(7601):73–76, 2016.

[38] Rampi Ramprasad, Rohit Batra, Ghanshyam Pilania, Arun Mannodi-Kanakkithodi, and Chiho Kim. Machine learning in materials informatics: recent applications and prospects. npj Computational Materials, 3(1):1–13, 2017.

[39] Victor Garcia Satorras, Emiel Hoogeboom, Fabian Bernd Fuchs, Ingmar Posner, and Max Welling. E (n) Equivariant Normalizing Flows. In Advances in Neural Information Processing Systems, 2021.

[40] Jonathan Schmidt, Love Pettersson, Claudio Verdozzi, Silvana Botti, and Miguel A. L. Marques. Crystal Graph Attention Networks for the Prediction of Stable Materials. Science Advances, 7 (49):eabi7948, 2021. doi: 10.1126/sciadv.abi7948.

[41] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A Continuous-filter Convolutional Neural Network for Modeling Quantum Interactions. Advances in Neural Information Processing Systems, 30, 2017.

[42] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. Journal of Machine Learning Research, 12(9), 2011.

[43] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. 2018.

[44] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A Deep Learning Approach to Antibiotic Discovery. Cell, 180(4):688–702, 2020.

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In International Conference on Learning Representations, 2018.

[46] Anthony Yu-Tung Wang, Steven K Kauwe, Ryan J Murdock, and Taylor D Sparks. Compositionally restricted attention-based network for materials property predictions. Npj Computational Materials, 7(1):1–10, 2021.

[47] Shiyu Wang, Xiaojie Guo, and Liang Zhao. Deep Generative Model for Periodic Graphs. arXiv preprint arXiv:2201.11932, 2022.

[48] Zhengyang Wang, Meng Liu, Youzhi Luo, Zhao Xu, Yaochen Xie, Limei Wang, Lei Cai, Qi Qi, Zhuoning Yuan, Tianbao Yang, et al. Advanced Graph and Sequence Neural Networks for Molecular Property Prediction and Drug Discovery. Bioinformatics, 38(9):2579–2586, 2022.

[49] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. npj Computational Materials, 2(1):1–7, 2016.

[50] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. Chemical science, 9(2):513–530, 2018.

[51] Tian Xie and Jeffrey C Grossman. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. Physical review letters, 120(14): 145301, 2018.

[52] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Crystal Diffusion Variational Autoencoder for Periodic Material Generation. In International Conference on Learning Representations, 2022.

[53] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Badly for Graph Representation? Advances in Neural Information Processing Systems, 34, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Conclusion and limitation section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix. A.2.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have included the detailed settings of our experiments. We will publish our implementation upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix. A.4 and Appendix. A.5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] Error bars are not necessary for baseline methods and our method.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] All of these are reported in Appendix. A.4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Cases breaking periodic invariance

We show two graph construction methods that break periodic invariance in Fig. 6 and Fig. 7.

One is the graph construction method employed by Graphormer in OC20 [3], shown in Fig. 6. It breaks periodic invariance because it treats every atom as a single node in the constructed graph. When the periodic boundaries are shifted, the inner structure can change a lot in a unit cell. Hence, the constructed fully connected graphs can be totally different when periodic boundaries are shifted.

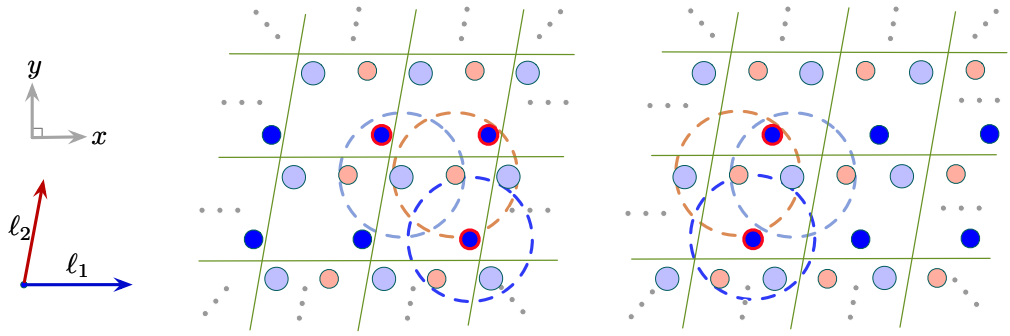


Figure 6: Illustration of graph construction method used by Graphormer [53] in OC20 [3]. We use orange, blue, and light blue to mark different atoms. We use circles of the same colors as atoms to denote the corresponding radius for atoms. It can be seen from the comparison of the left and the right that the constructed graphs are totally different when the periodic boundaries are shifted. In particular, the constructed graph on the left has three blue atoms, but the constructed graph on the right has two blue atoms. Thus, the graph construction method employed by Graphormer in OC20 breaks periodic invariance for crystals.

Another is the graph construction method using nearest neighbors based only on geometric pairwise distances, employed by CGCNN [51] and GATGNN [33], shown in Fig. 7. If several different atoms have the same geometric distances to the center atom, there is no deterministic way to determine which one to choose. Thus, for these cases, this method breaks periodic invariance.

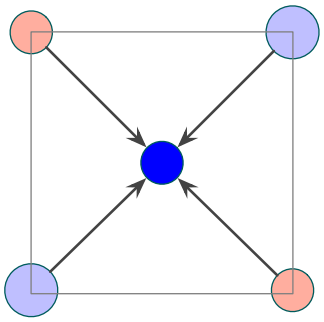


Figure 7: An illustration that crystal graph construction using nearest neighbors based only on pairwise distances breaks periodic invariance. We illustrate the one nearest neighbor case for simplicity. Black arrows denote the same pairwise distances from nearby atoms to the center blue atom. If sorting is based only on distances, there will be two different neighborhood formation approaches for the center blue atom, either using the light-blue atom or using the orange atom as the first nearest neighbor.

A.2 Proofs of periodic invariance

Notations. Recall that we use matrices $\mathbf{A}$, $\mathbf{P}$, and $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times 3}$ to describe a given crystal structure. $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n]^T \in \mathbb{R}^{n\times d_a}$ is the atom feature matrix, where $\boldsymbol{a}_i \in \mathbb{R}^{d_a}$ is the d_a-dimensional feature vector for atom i in the unit cell. $\mathbf{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n]^T \in \mathbb{R}^{n\times 3}$ is the position matrix, where $\boldsymbol{p}_i \in \mathbb{R}^3$ contains the Cartesian coordinates for atom i in 3D space. And the

infinite crystal structure can be represented as $\hat{\mathbf{P}} = \{\hat{\boldsymbol{p}}_i | \hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}, i \in \mathbb{Z}, 1 \leq i \leq n\}$, and $\hat{\mathbf{A}} = \{\hat{\boldsymbol{a}}_i | \hat{\boldsymbol{a}}_i = \boldsymbol{a}_i, i \in \mathbb{Z}, 1 \leq i \leq n\}$. Here, set $\hat{\mathbf{P}}$ contains all possible positions for each atom i, and set $\hat{\mathbf{A}}$ contains the corresponding atom feature vector for each atom i.

For crystal property prediction tasks, $\mathbf{L}$ representing the minimum repeating patterns is given. In the following proofs, we don't consider the case that the provided $\mathbf{L}' = \boldsymbol{\alpha}\mathbf{L}$ for $\boldsymbol{\alpha} \in \mathbb{N}_+^3$, which means the provided periodic patterns are not the minimum repeating patterns for a given crystal. When $\mathbf{L}$ representing the minimum repeating patterns is given, the periodic invariance lies in the invariance to shifts of periodic boundaries. We prove that the multi-edge graph construction and fully connected graph construction for crystals employed by our Matformer satisfy periodic invariance.

Multi-edge graph. The Multi-edge graph satisfies the periodic invariance by forming edges between node i and j using all items in the set $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}, d_{ij} \leq r\}$, where r is a prefixed threshold. It can be seen that all pairwise Euclidean distances between node i with positions $\{\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$, and node j with positions $\{\hat{\boldsymbol{p}}_j = \boldsymbol{p}_j + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$ are considered, and the shifts of periodic boundaries will not influence the pairwise Euclidean distances. Thus, this method satisfies periodic invariance. Additionally, by using pairwise Euclidean distances, unit cell E(3) invariance is naturally satisfied.

We note that for a given node i, the radius can be computed by a deterministic function that takes all pairwise distances between node i and all other nodes as input, and produces a real value r as output. This will not break periodic invariance due to the fact that the deterministic function will always produce the same r for the same input $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}, 1 \leq j \leq n\}$. One example is that we can use the 12-th smallest distance d_{12} in $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}, 1 \leq j \leq n\}$ as the radius for node i.

Fully connected graph for crystals. To construct fully connected graphs for crystals, in our design, node i represents atom with atom feature $\mathbf{a}_i$ and all its repeats in the infinite 3D space, with positions $\{\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$. In the t-fully-connected graph for crystals, for node i and j, t smallest pairwise distances in $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}\}$ are considered, where t is a hyperparameter to control the edge number. Similar to the proof of multi-edge graph, all pairwise Euclidean distances between node i with positions $\{\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$, and node j with positions $\{\hat{\boldsymbol{p}}_j = \boldsymbol{p}_j + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$ are considered, and the shifts of periodic boundaries will not influence the pairwise Euclidean distances. Hence, this method satisfies periodic invariance. In addition, the usage of pairwise Euclidean distances makes the unit cell E(3) invariance naturally satisfied.

A.3 Dataset descriptions

We show the detailed dataset Descriptions for two crystal datasets used in experimental studies, including The Materials Project and JARVIS.

The Materials Project dataset. In particular, for tasks of formation energy and band gap, we directly follow ALIGNN [6] and use the same training, validation, and test set, including 60000, 5000, and 4239 crystals, respectively. For tasks of Bulk Moduli and Shear Moduli, we follow GATGNN [33], the recent state-of-the-art method for these two tasks, and use the same training, validation, and test sets, including 4664, 393, and 393 crystals. In Shear Moduli, one validation sample is removed because of the negative GPa value. We either directly use the publicly available codes from the authors, or re-implement models based on their official codes and configurations to produce the results. Detailed configurations of these retrained models are provided in Appendix. A.5.

The JARVIS dataset. JARVIS is a newly released database proposed by Choudhary et al. [8]. For JARVIS dataset, we follow ALIGNN [6] and use the same training, validation, and test set. We evaluate our Matformer on five important crystal property tasks, including formation energy, bandgap(OPT), bandgap(MBJ), total energy, and Ehull. The training, validation, and test set contains 44578, 5572, and 5572 crystals for tasks of formation energy, total energy, and bandgap(OPT). The numbers are 44296, 5537, 5537 for Ehull, and 14537, 1817, 1817 for bandgap(MBJ). The used metric is test MAE. The results for CGCNN and CFID are taken from ALIGNN [6], other baseline results are obtained by retrained models. Detailed configurations of these retrained models are provided in Appendix. A.5.

A.4 Matformer configurations

We show the detailed configurations of our Matformer models in this section.

Notations. $\boldsymbol{a}_i \in \mathbf{A}$ is the d_a-dimensional feature vector for node i, $\boldsymbol{e}_{ij}^h \in \mathbf{E}$ is d_e-dimensional feature vector for the h-th edge between nodes i and j, and $\boldsymbol{f}_i^\star$ is the input feature vector of node i for a given layer of Matformer. LN_Q, LN_K, and LN_E denote the linear transformations to compute query, key, and edge embedding in a given Matformer layer, respectively. $\boldsymbol{q}_i$, $\boldsymbol{k}_i$ and $\boldsymbol{v}_i$ are the computed query, key, and value vectors after these linear transformations for node i. $\boldsymbol{e}_{ij}^{h'}$ is the intermediate output in a given Matformer layer for $\boldsymbol{e}_{ij}^h$.

Crystal graph construction. For all Matformer models used in two material datasets, including The Materials Project and JARVIS, we use the radius-based multi-edge graph construction method. We use the 12-th smallest distance between the given atom and all nearby atoms as radius, and include all nearby atoms within the radius as the neighborhood for the given atom. It satisfies periodic invariance as described in Appendix. A.2. For the encoding of periodic patterns, we use six geometric distances, including $||\boldsymbol{\ell}_1||_2$, $||\boldsymbol{\ell}_2||_2$, $||\boldsymbol{\ell}_3||_2$, $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2||_2$, $||\boldsymbol{\ell}_2 + \boldsymbol{\ell}_3||_2$, and $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_3||_2$ in our study. For any of these distances, if it is already in the radius of node i computed by the previous graph construction method, it will not be added as a self-connecting edge of node i in the final graph.

Node and edge embeddings. For each node, we map the atomic number to a 92-dimensional embedding using CGCNN [51] atomic embedding. We then use a linear transformation to map it to a 128-dimensional vector as the input $\boldsymbol{f}_i^*$ to the first Matformer message passing layer. For each edge, we map the Euclidean distance to a 128-dimensional embedding using 128 RBF kernels with centers from 0.0 to 8.0. It is then mapped to a 128-dimensional vector as the edge input $\boldsymbol{e}_{ij}^h$, by a nonlinear layer followed by a linear layer.

Matformer layer. For message passing, we use five layers of Matformer Message Passing layer, with four attention heads. The embedding sizes for $\boldsymbol{q}_i$, $\boldsymbol{k}_i$, $\boldsymbol{v}_i$ for a single head are 128, mapped from $\boldsymbol{f}_i^*$ using corresponding linear transformations LN_Q, LN_K, and LN_V. For a given layer, different heads use different LN_Q, LN_K, LN_V, and LN_E, but share other operations. To obtain the final message m_i, the features from four heads are concatenated and mapped to a 128-dimensional vector by a linear transformation.

Readout layer. We use the mean pooling to aggregate features from all nodes in a graph and then use a nonlinear layer with hidden dimension 128 followed by a linear layer to obtain the scalar output for a crystal graph.

Training hyperparameters. For all tasks in The Materials Project and JARVIS, we use the Adam optimizer [21] with weight decay [32] of 1e-5 and one cycle learning rate scheduler [43]. We use the batch size of 64. We use mean square error as the objective function to train and mean absolute error as the evaluation metric to validate and test. We only slightly adjust the learning rates and training epochs for different tasks in The Materials Project and JARVIS, as shown in Table. 7 and Table. 8, respectively. We use Pytorch to implement our models. For all tasks on two benchmark datasets, we use one NVIDIA RTX A6000 48GB GPU to train our Matformer models.

Table 7: Training hyperparameters for Matformer models on The Materials Project.

Parameter	Formation Energy	Bandgap	Bulk Moduli	Shear Moduli
Learning rate	1e-3	5e-4	1e-3	1e-3
Epoch number	500	500	500	300

Table 8: Training hyperparameters for Matformer models on JARVIS.

Parameter	Formation Energy	Bandgap(OPT)	Total Energy	Ehull	Bandgap(MBJ)
Learning rate	1e-3	8e-4	1e-3	1e-3	1e-3
Epoch number	500	300	500	500	300

We also show the detailed Matformer architecture in Fig. 5.

Table 9: Comparison in terms of test MAE on The Materials Project dataset. We show results both from retrained models and referred papers to make the comparison clear and fair. Results from original papers are shown in parentheses () on the right. '-' denotes no results are reported in referred papers. The best results are shown in **bold** and the second best results are shown with underlines.

	Formation Energy	Band Gap	Bulk Moduli	Shear Moduli
Method	eV/atom	eV	log(GPa)	log(GPa)
CGCNN [51]	0.031 (0.039)	0.292 (0.388)	0.047 (0.054)	0.077 (0.087)
SchNet [41]	0.033 (0.035)	0.345 (-)	0.066 (-)	0.099 (-)
MEGNET [4]	0.030 (0.028)	0.307 (0.33)	0.060 (0.050)	0.099 (0.079)
GATGNN [33]	0.033 (0.039)	0.280 (0.31)	0.045	0.075
ALIGNN [6]	0.022	0.218	0.051 (-)	0.078 (-)
Matformer	**0.021**	**0.211**	**0.043**	**0.073**

A.5 Configurations of retrained models

We show configurations of retrained models for The Materials Project and JARVIS in this section. The results from their original papers are shown in Table. 9.

SchNet [41]. We use six layers of SchNet message passing layer following the original paper, with feature dimension of 64. We train SchNet on these four tasks with learning rate of 5e-4 and batch size of 64 for 500 epochs. The Adam optimizer is used with 1e-5 weight decay. One cycle learning rate scheduler is also used. We use the 12-th smallest distance between the given atom and all nearby atoms to serve as the radius for this given atom for the tasks of formation energy and bandgap in The Materials Project and all tasks in JARVIS. For the tasks of shear moduli and bulk moduli, We use the 32-th smallest distance between the given atom and all nearby atoms to serve as the radius for every atom to better the performance.

MEGNET [4]. Following the original paper, we use three layers of MEGNET message passing layer with the same feature dimensions as mentioned in the paper, and use Set2Set readout function. We train MEGNET on these four tasks with learning rate of 1e-3 and batch size of 128 for 1000 epochs following the configuration settings mentioned in the original paper. The Adam optimizer is used with 1e-5 weight decay. One cycle learning rate scheduler is also used. We try different hyperparameters of crystal graph construction and batch size to better the performances of MEGNET. In particular, for formation energy and bandgap, we use a radius of 4.0 for all atoms with batch size of 128. And for bulk moduli and shear moduli and all tasks in JARVIS, we use the 12-th smallest distance between the given atom and all nearby atoms to serve as the radius for this given atom to better the performance, and the batch size of 64 is chosen because of better empirical results.

CGCNN [51]. For CGCNN, we directly use the publicly available code from Xie and Grossman [51], with 128 hidden dimensions, batch size of 256, and three layers of CGCNN message passing layer. We train CGCNN with Adam optimizer because of better empirical results. We train CGCNN for these four tasks using learning rate of 1e-2 for 100 epochs and 1e-3 for the next 900 epochs following the official code. The radius cutoff of 8.0 is used for all atoms, and the nearest 12 neighbors are selected.

GATGNN [33]. For GATGNN, we directly use the publicly available code from Louis et al. [33], with 128 hidden dimensions, batch size of 256 and keep other default settings. We train GATGNN with learning rate of 5e-3 with learning rate decay milestone of 300 epochs and decay parameter of 0.5. We train GATGNN for 500 epochs for the task of formation energy and bandgap in The Materials Project and all tasks in JARVIS, with early stop. The radius cutoff of 4.0 is used for all atoms, and the nearest 16 neighbors are selected according to the original paper.

ALIGNN [6]. For ALIGNN, we directly use the publicly available code from Choudhary and DeCost [6]. We use the official best model configurations of ALIGNN to train ALIGNN models on the tasks of bulk moduli and shear moduli, with learning rate of 1e-3 and batch size of 64. In particular, we use ALIGNN with four gcn layers and four alignn layers.

Overall, we either directly use the publicly available codes from corresponding authors [51, 33, 6], or re-implement models based on their official codes and configurations [41, 4] to produce the results in our experiments.

A.6 Building block of Matformer

We provide ablation studies about sigmoid and layernorm instead of softmax in our Matformer message passing layer in this section.

We evaluate the operation of sigmoid and normalization instead of softmax in our message passing, which is designed to capture the node degree information in multi-edge crystal graphs, described in Sec. 4.2. Compared with using softmax with scalar attention coefficient, denoted as Softmax-scalar, or softmax with vector attention coefficient, denoted as Softmax-vector, using sigmoid and normalization leads to significant performance improvement in terms of test MAE, justifying the effectiveness of this operation in Matformer message passing design, as shown in Table. 10.

Table 10: Operation ablation study.

Operation	Test MAE
Softmax-scalar	0.0376
Softmax-vector	0.0347
Sigmoid and Norm	**0.0325**

A.7 Complexity analysis of introducing angular information

We provide the complexity analysis of adding angular information to Matformer in this section.

Assume that we have n atoms in a single cell and thus n nodes in the original multi-edge graph. Also assume that every node has at least 12 neighbors following the graph construction method of Matformer and assume there are no self-connecting edges. This will result in a graph $G = (V, E)$ where $|V| = n$ and $|E| = 12/2n = 6n$. When converting graph G into line graph $L(G)$, every edge is treated as a node in the line graph. So we have $6n$ nodes in the line graph. Every edge in the original graph is connecting 2 nodes and every node has $(12 - 1)$ other edges, resulting in 22 neighboring edges for each edge in the original graph. So we have $22 * 6n/2 = 66n$ edges in the converted line graph. Compared with original graph with $|V| = n$ and $|E| = 6n$, the converted line graph is super large with node number of $6n$ and edge number of $66n$.