Efficient determination of the Hamiltonian and electronic properties using graph neural network with complete local coordinates

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
12 January 2023

REVISED
28 March 2023

ACCEPTED FOR PUBLICATION
6 April 2023

PUBLISHED
24 July 2023

PAPER

Efficient determination of the Hamiltonian and electronic properties using graph neural network with complete local coordinates

Mao Su[1,3], **Ji-Hui Yang**[1,2], **Hong-Jun Xiang**[1,2] and **Xin-Gao Gong**[1,2,*]

[1] Key Laboratory for Computational Physical Sciences (MOE), State Key Laboratory of Surface Physics, Department of Physics, Fudan University, Shanghai 200433, People's Republic of China
[2] Shanghai Qi Zhi Institute, Shanghai 200030, People's Republic of China
[3] Shanghai AI Laboratory, Shanghai 200030, People's Republic of China
[*] Author to whom any correspondence should be addressed.

E-mail: xggong@fudan.edu.cn

Keywords: Hamiltonian, neural network, local coordinate

Abstract

Despite the successes of machine learning methods in physical sciences, the prediction of the Hamiltonian, and thus the electronic properties, is still unsatisfactory. Based on graph neural network (NN) architecture, we present an extendable NN model to determine the Hamiltonian from *ab initio* data, with only local atomic structures as inputs. The rotational equivariance of the Hamiltonian is achieved by our complete local coordinates (LCs). The LC information, encoded using a convolutional NN and designed to preserve Hermitian symmetry, is used to map hopping parameters onto local structures. We demonstrate the performance of our model using graphene and SiGe random alloys as examples. We show that our NN model, although trained using small-size systems, can predict the Hamiltonian, as well as electronic properties such as band structures and densities of states for large-size systems within the *ab initio* accuracy, justifying its extensibility. In combination with the high efficiency of our model, which takes only seconds to get the Hamiltonian of a 1728-atom system, the present work provides a general framework to predict electronic properties efficiently and accurately, which provides new insights into computational physics and will accelerate the research for large-scale materials.

1. Introduction

The Hamiltonian lies at the heart of solid-state physics as it determines the electronic properties, the eigenstates of a Hamiltonian can then be used to obtain other physical properties. For periodic systems, the Hamiltonian needs to be solved in reciprocal space, which then gives band structure information. The past half-century has witnessed the development of solving Hamiltonian problems based on density functional theory (DFT) [1], which has achieved great success in understanding electronic properties. However, with increasing system size, the cost of solving Hamiltonian using DFT increases dramatically and has become one of the greatest challenges in solid-state physics and materials science.

Machine learning can provide a possibility to solve such problems. However, despite past decades having witnessed many successes of machine learning methods in predicting physical properties [2, 3] and interatomic potentials [4–6], prediction of the Hamiltonian, and thus electronic properties, is still unsatisfactory. Current models generally yield low accuracy and poor extensibility to predict electronic properties such as bandgaps, defects, and optical properties. Mapping the Hamiltonian onto some locality is crucial to design an extendable model [7, 8]. Inspired by the tight-binding method, the Hamiltonian can be represented by hopping parameters connecting the localized basis functions. Using numerical pseudo-atomic orbitals (PAOs) ϕ $(\mathbf{r})$ as the basis [9], the hopping parameter between atoms i and j is defined by

$$h_{i\alpha,j\beta}^{(\mathbf{R}_n)} = \left\langle \phi_{i\alpha}\left(\mathbf{r}-\boldsymbol{\tau}_i\right)\middle|\hat{H}\middle|\phi_{j\beta}\left(\mathbf{r}-\boldsymbol{\tau}_j-\mathbf{R}_n\right)\right\rangle, \quad (1)$$

where α and β denote the orbitals, τ is the atomic position, and $\mathbf{R}_n$ is the lattice vector. Note that hopping parameters have locality and can be determined by local structures because hopping can be negligible when atomic distance is larger than a certain cutoff radius [7]. Therefore, the Hamiltonian matrix, which is constructed by hopping parameters, can be determined from local structures, and thus an extendable model for predicting the Hamiltonian is theoretically feasible. In previous studies, Hegde and Bowen proposed a model for predicting Hamiltonians in a small basis set and studied the s-orbital Cu and sp3-orbital diamond systems [10]. Schütt *et al* developed the SchNOrb model for small, individual molecules with data augmentation [11]. Gu *et al* introduced a neural network (NN) representation of tight-binding Hamiltonians for one-dimensional carbon chains [12]. The main disadvantage of the above methods is that they did not treat the equivariance appropriately, and thus the applications are limited.

Because the Hamiltonian matrix is equivariant under rotations of coordinates, the prediction of hopping parameters is more complicated than predicting measurable properties such as bandgaps, which are independent of the choice of coordinate system. Although the data augmentation technique can reflect rotations [11], it is obviously inefficient. To fix the rotation freedoms, Li *et al* recently introduced the deepH model [13], in which the local coordinate (LC) for each atom pair is defined based on the local structures. The hopping parameters can be trained in the LC and then transformed back to the global coordinate to construct the Hamiltonian matrix. Using the atomic-like basis functions, the rotational transformations can be well described by the Wigner-D matrix. Nevertheless, more efficient methods for predicting the Hamiltonian are still being explored. For example, in the work of Li *et al*, the hermiticity of the Hamiltonian is not properly considered. Moreover, the commonly defined LC is an incomplete descriptor, and this will be clarified in the following.

In this work, based on the graph NN architecture and LC transformation, we have developed an extendable model, called graph LC-Net, to predict the hopping parameters and hence obtain DFT Hamiltonian. In addition to the rotational covariance, the key advantages of our model over similar works are as follows: the hermiticity of the Hamiltonian is utilized, a complete LC for each atom pair is defined by our new algorithm to avoid the degeneration problem for highly symmetric structures and the convolutional neural network (CNN) is used to handle the LC information. The node and edge features are updated by message passing with an attention mechanism. The output of the model is the triangular portion of the hopping parameter matrix, and then all the hopping parameters are assembled into the Hamiltonian matrix. Using graphene and SiGe random alloy systems as examples, we demonstrate that our graph LC-Net, trained using small-size systems, can predict the Hamiltonian as well as electronic properties such as band structures and densities of states (DOSs) for large-size systems within the DFT accuracy, but with very little cost, i.e. it takes only about 10 s to get the Hamiltonian of a 1728-atom system using only one GPU card. In addition, the linear scaling with the number of atoms of graph LC-Net makes it very promising for studying the electronic properties of large-scale systems.

2. Results

2.1. Preliminary

Graph is a data structure composed of sets of vertices and edges. The atomic structure is represented by a graph, in which the nodes represent the atoms, and the edges represent the bonds between atoms. In the tight-binding model, the hopping parameters are determined by the local structures of atom pairs. Therefore, the task in this work is to predict the properties of edges. Message passing NNs [14] is a universal graph neural network (GNN) framework proposed for graph, where the node features are updated by gathering the features of all the neighbors. The most straightforward way to update the edge features is to aggregate the features of the two nodes on the edge. GNN is widely used to model interatomic potentials [15], predict materials properties [16], and drug design [17].

2.2. Embedding layer

As shown in figure 1, the graph LC-Net model consists of an embedding layer, an LC layer, several interaction layers, and an output layer. The inputs include the atomic number $\{Z_i\}$, the edge distances $\{|r_{ij}|\}$, and the LC information $\{y_{ij}\}$. In the embedding layer, each node is encoded as a one-hot vector according to the atomic number. The connectivity of the graph is determined by the distances between nodes, that is, there is an edge if the distance between the two nodes is less than the cutoff radius R_c. The initial node features $v_{ij}^{(0)} \in \mathbb{R}^{N_v}$ are calculated through an embedding network [15] by $v_i^{(0)} = a_{Z_i}$. The initial edge features

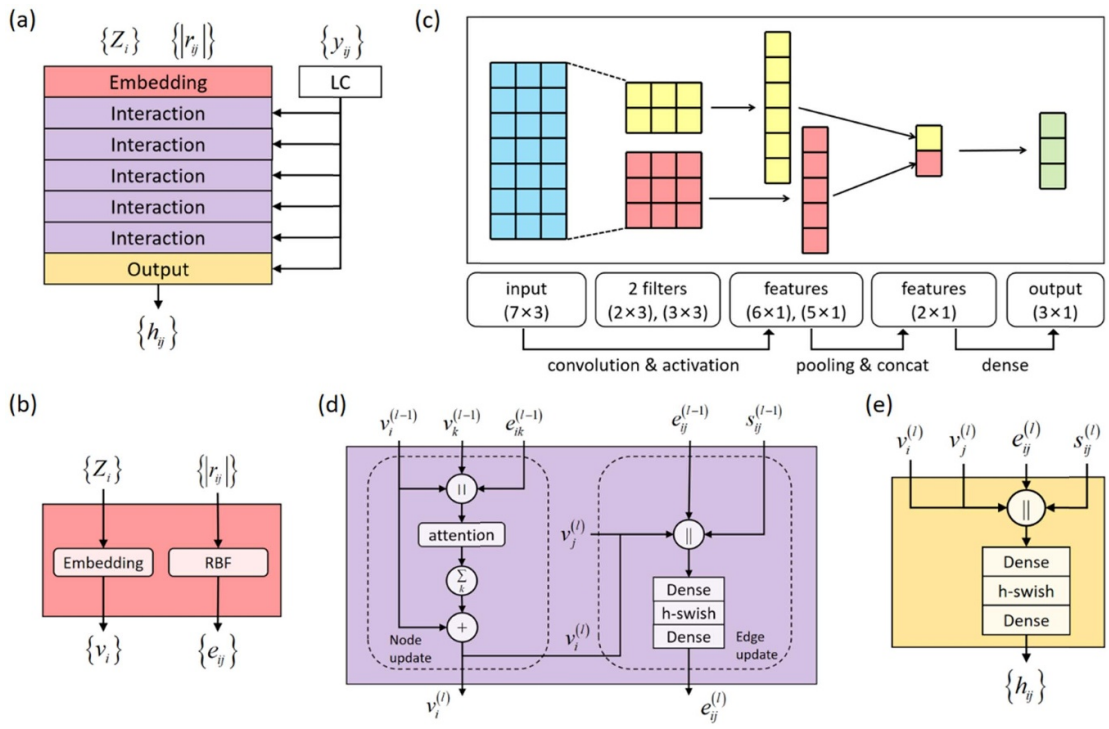

Figure 1. The architecture of graph LC-Net. (a) Graph LC-Net consists of an embedding layer, an LC layer, five interaction layers, and an output layer. The output of the LC layer is passed to each interaction layer. (b) Embedding layer. The atomic numbers are transformed into the initial node features through an embedding network. The interatomic distances are expanded by Gaussian radial basis functions to generate initial edge features. (c) An example LC layer with two filters. Spherical harmonics are used to generate LC features. (d) Interaction layer. The node features and edge features are updated using message passing. (e) Output layer. The node, edge, and LC features are combined to output hopping parameters.

$e_{ij}^{(0)} \in \mathbb{R}^{N_e}$ are the Gaussian radial basis functions $e_{ij}^{(0)} = \exp\left(-\left(|r_{ij}| - r_n\right)^2 \Big/ \sigma^2\right)$, where the centers r_n are placed linearly between 0 and R_c, and σ^2 is the variance.

2.3. LC layer

The LC layer is used to obtain the LC information for each edge e_{ij} which is uniquely determined by the local structures of atoms i and j. Note that, to utilize the hermiticity of the Hamiltonian, the edges ij and ji must be in the same LC. The details of calculating LC are described in the methods section. The LC information can be represented by real spherical harmonics and distances. Here we use the edge ij as an example to illustrate how LC information is obtained. First, the neighbors of atom i, denoted by k, are sorted by distances $|r_{ik}|$. Then we calculate the real spherical harmonics of bond ik, denoted by $Y_{lm}\left(\theta_{ik}^{ij}, \varphi_{ik}^{ij}\right)$, where θ_{ik}^{ij} and φ_{ik}^{ij} are the polar and azimuthal angles, respectively, of bond ik relative to an LC defined for edge ij. Each bond ik can be represented by a list of real spherical harmonics, yielding the orientational vector $y_{ik}^{ij} \in \mathbb{R}^{N_i \times N_y}$ calculated by $y_{ik}^{ij} = Y_{lm}\left(\theta_{ik}^{ij}, \varphi_{ik}^{ij}\right) \Big/ \left(1 + |r_{ik}|\right)$, where N_i is the number of neighbors of atom i and N_y is the number of spherical harmonic functions. Similarly, for atom j we can obtain the orientational vector $y_{jk}^{ij} \in \mathbb{R}^{N_j \times N_y}$.

To extract the LC features s_{ij} from the orientational vectors y_{ik}^{ij} and y_{jk}^{ij}, Li *et al* introduce the LC message passing (LCMP) layer [13]. However, the order of the neighbors is not considered in the LCMP layer. To preserve the information of neighbor order, we utilize the CNN architecture for natural language processing (text-CNN) [18, 19] as shown in figure 1(c). The convolution operator with a filter $w \in \mathbb{R}^{h \times N_y}$ is applied to a window of h neighbors to produce a feature. Using the convolution operator on the input y_{ik}^{ij} generates a new feature map $c \in \mathbb{R}^{N_i - h + 1}$. Then max pooling is performed to extract a scalar from each feature map. These features are then passed to a dense layer to generate the output vector $s_i \in \mathbb{R}^{N_{LC}}$. The neighbors of atom j are processed similarly, and finally, the outputs are concatenated to give the LC feature vector $s_{ij} \in \mathbb{R}^{N_{LC} \times 2}$.

We use an example to further illustrate how to utilize the text-CNN to extract LC information. Consider a local structure with respect to atom i in figure 2(a), the LC is defined on edge ij, and atoms 1–4 are neighbors of atom i. The input of the LC layer is the spherical harmonics $Y_{lm}^{i1}, \ldots, Y_{lm}^{i4}$. Each Y_{lm} is a vector of

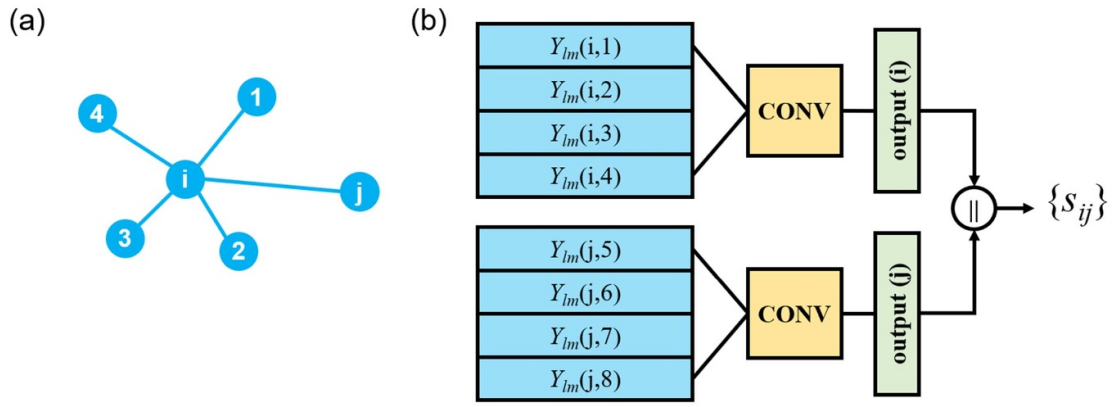

Figure 2. Illustration of using CNN to extract LC information. (a) The local structure of atom *i*. (b) Schematic of extracting the LC information of edge *ij*. The atoms 1–4 are neighbors of atom *i* and the atoms 5–8 are neighbors of atom *j*.

$N_y = 25$ elements as $0 \leqslant l \leqslant 4$ and $-l \leqslant m \leqslant l$ are used. In analogy with the text-CNN framework, each neighbor is treated as a word in our model. Since the neighbors are sorted, we expect that the NN is able to learn the information of order, which is not considered by the LCMP [13]. Nevertheless, the text-CNN layer could handle the serialized information appropriately. When a different LC is chosen, the input of spherical harmonics will be different, and thus the information of LC is learned. We have used three windows of sizes 2, 3 and 4, each of which has 64 filters. Thus, for each LC, 192 new features are extracted by the convolution operator. The output is a vector of N_{LC} elements representing the LC information from the neighbors of atom *i*. The neighbors of atoms *j* are processed similarly. Finally, the outputs of atom *i* and atom *j* are concatenated to give LC feature s_{ij} as shown in figure 2(b).

2.4. Interaction layer

After the processing of the embedding layer and the LC layer, information is passed to the multiple-stacked interaction layers. In the interaction layer, the attention mechanism [20, 21] is used for updating node features. To alleviate the over-smoothing problem, we have used skip connections inspired by ResNet [22]. Note that, our model is a universal framework, the interaction layers could be replaced by other advanced graph-based models for potentially better performance. To utilize the LC features, we just concatenate the LC feature s_{ij} and the edge feature e_{ij}, since they are in one-to-one correspondence. As shown in figure 1(d), the node features $v_i \in \mathbb{R}^{N_v}$ are updated by

$$v_i^{(l)} = \sum_{k \in \mathcal{N}(i)} \alpha_{i,k} \Phi v_k^{(l-1)} + v_i^{(l-1)}, \tag{2}$$

where *k* denotes the neighbor of *i*, *l* refers to the *l*-th interaction layer, and Φ denotes a dense layer. The attention coefficients $\alpha_{i,k}$ are computed from

$$\alpha_{i,k} = \frac{\exp\left(a^T \mathrm{LeakyReLU}\left(\Phi\left[v_i || v_k || e_{ik} || s_{ik}\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(a^T \mathrm{LeakyReLU}\left(\Phi\left[v_i || v_k || e_{ik} || s_{ik}\right]\right)\right)}, \tag{3}$$

where a^T is a learnable weight variable, LeakyReLU [23] is an activation function and $||$ denotes concatenation. After the node features are updated, the edge features are updated from

$$e_{ij}^{(l)} = \Phi\left[\text{h-swish}\left[\Phi\left(v_i^{(l)} || v_j^{(l)} || e_{ij}^{(l-1)} || s_{ij}\right)\right]\right], \tag{4}$$

where h-swish is the hard version of the swish activation function [24].

2.5. Output layer

The output layer consists of two stacked dense layers with h-swish,

$$h_{ij} = \Phi\left[\text{h-swish}\left[\Phi\left(v_i || v_j || e_{ij} || s_{ij}\right)\right]\right], \tag{5}$$

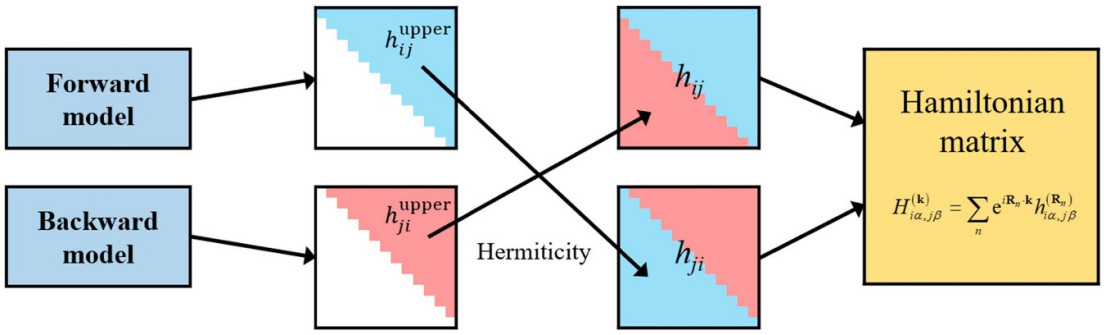

Figure 3. The hopping parameters are divided into two groups according to the definition of LC. We train a forward model for the group $\{h_{ij}\}$ and a backward model for the group $\{h_{ji}\}$ respectively. The two models share the same architectures and input information. The forward model outputs the upper triangular part with the main diagonal of $\{h_{ij}\}$, and the backward model outputs the upper triangular part without the main diagonal of $\{h_{ji}\}$. The lower triangular parts are determined by hermiticity. Then all the hopping parameters are assembled into the Hamiltonian matrix.

where $h_{ij} \in \mathbb{R}^{N_h}$ is the upper triangular portion of the hopping matrix of ij. Since the Hamiltonian is Hermitian, the model does not need to output all matrix elements. However, the hopping matrix in general is not Hermitian, that is, $h_{i\alpha,j\beta} \neq h^*_{i\beta,j\alpha}$ unless $i = j$. Considering the hermiticity of the Hamiltonian, it can be shown that the hopping matrix satisfies $h_{i\alpha,j\beta} = h^*_{j\beta,i\alpha}$. Therefore, only the upper (or lower) triangular portion of hopping parameters are required as the output of the model. There is a problem that, in general, the upper triangular portion of h_{ij} and h_{ji} are different, that is, $h_{ij} \neq h_{ji}$. The hermiticity can be utilized only if h_{ij} and h_{ji} are in the same LC. In this case, the input information for predicting h_{ij} and h_{ji} are the same, that is, $h_{ij}^{\text{Pred}} = h_{ji}^{\text{Pred}}$, and then the training will not converge. To overcome this problem, the hopping matrices are divided into two groups according to the distances of the neighbors, as described in the method section. The two groups are trained separately. Then we obtain two models for the hopping parameters, called the *forward model* and the *backward model*, respectively. Finally, the Hamiltonian matrix is constructed by all the hopping matrices, as shown in figure 3.

2.6. Loss function

The graph LC-Net is trained by minimizing the difference between the predicted data h_{ij} and the DFT calculated data $\hat{h}_{ij}$. We note that the hopping parameters are highly imbalanced, that is, most targets are close to zero while a few are larger than 10 eV. To better handle the imbalanced data, a regression version of the focal loss [25, 26] is used with

$$L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\beta|y_i-\hat{y}_i|}}\right)^{\gamma}|y_i-\hat{y}_i|, \tag{6}$$

where β and γ are adjustable hyper-parameters, and y denotes the output hopping parameters.

2.7. Model evaluation

With the above architecture for learning the Hamiltonian in hand, we turn to real systems to demonstrate the feasibility of learning electronic properties using the NN methods. The goal is to map the local atomic structures to the hopping parameters $h_{i\alpha,j\beta}^{(\mathbf{R}_n)}$ using graph LC-Net. In this work, s2p2d1 PAOs are used as basis functions and the hopping parameters for each atom pair ij are represented by a 13×13 matrix. The mean absolute error (MAE) between DFT calculated and NN hopping parameters is used to evaluate our method. The electronic properties including band structures and DOSs are then calculated for comparison.

2.8. Degeneration problem with local symmetry

The first step of training a Hamiltonian model is to define the LC for each atom pair. In the previous study [13], the LC is determined by the nearest neighbor n_1 and the second nearest neighbor n_2. However, this method only works on perturbated structures, that is, the nearest and second-nearest neighbors are unique. In the case of highly symmetric local structures, for instance, defect-free crystals, the above method fails to determine a unique LC since the nearest neighbor cannot be determined uniquely by distances, which then results in data conflict. To be specific, for the neighbors of the atom i, if $|\mathbf{r}_{n_1} - \mathbf{r}_i| = |\mathbf{r}_{n_2} - \mathbf{r}_i|$, the atoms n_1

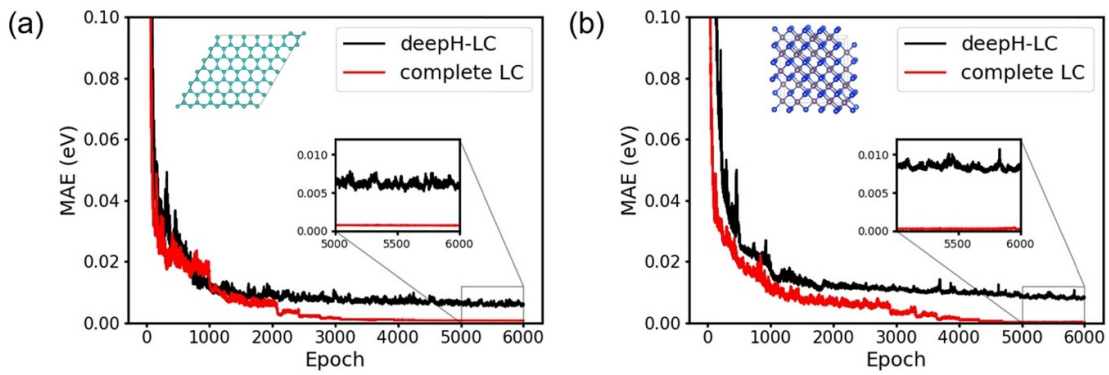

Figure 4. The learning curve of predicting the hopping parameters with different methods for LC. (a) An example of a perfect graphene structure. (b) An example of a zincblende SiGe alloy. Black line: the LCs are calculated using the method of deepH. Red line: the complete LCs are calculated using the algorithm proposed in this work.

and n_2 are indistinguishable. This is also known as the *degeneration problem* [27]. Similar problems can happen for the off-site hopping parameters. As a result, the errors of highly symmetric structures will be significantly larger, even in the training set, than those of randomly perturbated structures due to data conflict. We find that this problem can be solved by utilizing the information of the global coordinate to define a *complete* LC for each edge, and the algorithm is provided in the method section.

To illustrate the feasibility of the new algorithm, we test the performance of different methods on a perfect graphene structure and a zincblende SiGe alloy. We first calculate LC for each edge using the original method of deepH [13] and then perform calculations with our new algorithm. The learning curves for graphene and SiGe are shown in figures 4(a) and (b), respectively. It is shown that the training loss of the model with the original LC is significantly larger than that with our complete LC.

2.9. Graphene dataset

We create a dataset of graphene to compare the performance of our graph LC-Net with the DeepH model proposed in [13]. We perform molecular dynamics simulation of a $6 \times 6 \times 1$ graphene structure for 5 ps to generate the dataset, and the computational details are consistent with the case of DeepH. Then we sample 500 structures as the training set and the other 500 structures as the validation set. We find that, on the validation set, the MAEs of the forward model and the backward model are 3.5 meV and 1.9 meV, respectively, as shown in figures 5(a) and (b). Our result is comparable with that of DeepH (0.4 meV–8.5 meV). The element-wise error distributions are shown in figure 5(c). The band structures and DOS are shown in figure 5(d). Note that, the authors of DeepH train 169 models for the hopping parameters to reconstruct the Hamiltonian of the graphene dataset, while we train only 2 models to obtain all the hopping parameters, demonstrating the efficiency of our method.

2.10. SiGe random alloy dataset

We demonstrate the feasibility of our method with the three-dimensional SiGe random alloy. The SiGe random alloy dataset is generated by randomly occupying the zinc-blende lattice sites with the Si or Ge atoms. The number of possible combinations in a supercell with N sites is given by the combinatorial number $C(N, N/2)$, which could be incredibly large as the total atom number increases. Note that, most of the structures are identical under certain symmetry operations. Therefore, we develop a scheme to filter the identical structures to construct a training set of SiGe random alloy.

For each atom, we construct a unique local structure information by its neighbors in the following way. As shown in figure 6(a), we first calculate the dot product of $\mathbf{r}_{A,0}$ and $\mathbf{r}_{B,0}$(atomic positions with respect to the target atom 0), and then obtain a list of dot products for the target atom:

$$v(0) = [\mathbf{r}_{10} \cdot \mathbf{r}_{20}, \mathbf{r}_{10} \cdot \mathbf{r}_{30}, \mathbf{r}_{10} \cdot \mathbf{r}_{40}]. \tag{7}$$

Then we sort the elements in the list $v(0)$ in ascending order to represent the local structure information. It is easy to check that the two local structures in figure 6(a) are different by comparing the lists.

We calculate the local structure encodings by equation (7) for all atoms, remove duplicated ones and add the encodings into a database. When a new structure is generated, we compare the new encodings with those in the database. If new local structures are found, the generated structure is added to the training set and the database is updated accordingly. New structures are generated repeatedly until no more new local structures

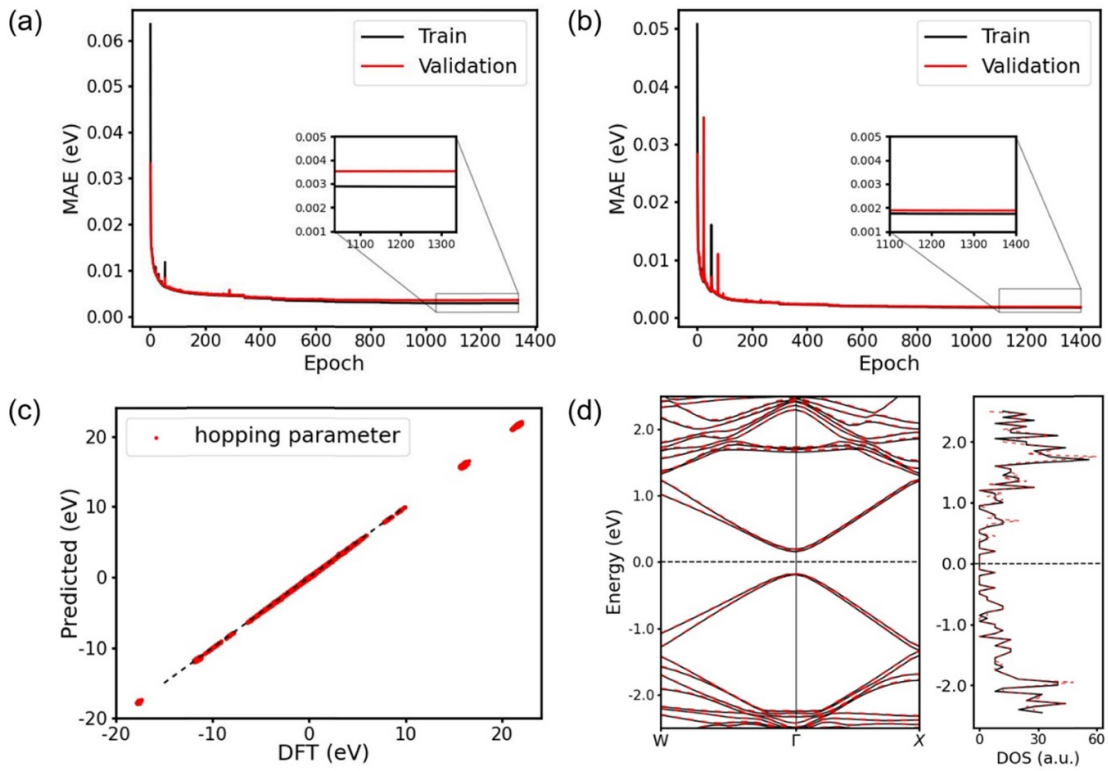

Figure 5. Model evaluation with the 6 × 6 × 1 graphene dataset. (a) The learning curve of the forward model. The LCs are defined by equations (13) and (14). (b) The learning curve of the backward model. The LCs are defined by equations (13) and (15). (c) Element-wise error distributions of the hopping parameters of a graphene in the validation set. (d) Band structure and DOS of a graphene in the validation set. Black line: results of graph LC-Net. Red dotted line: DFT results.

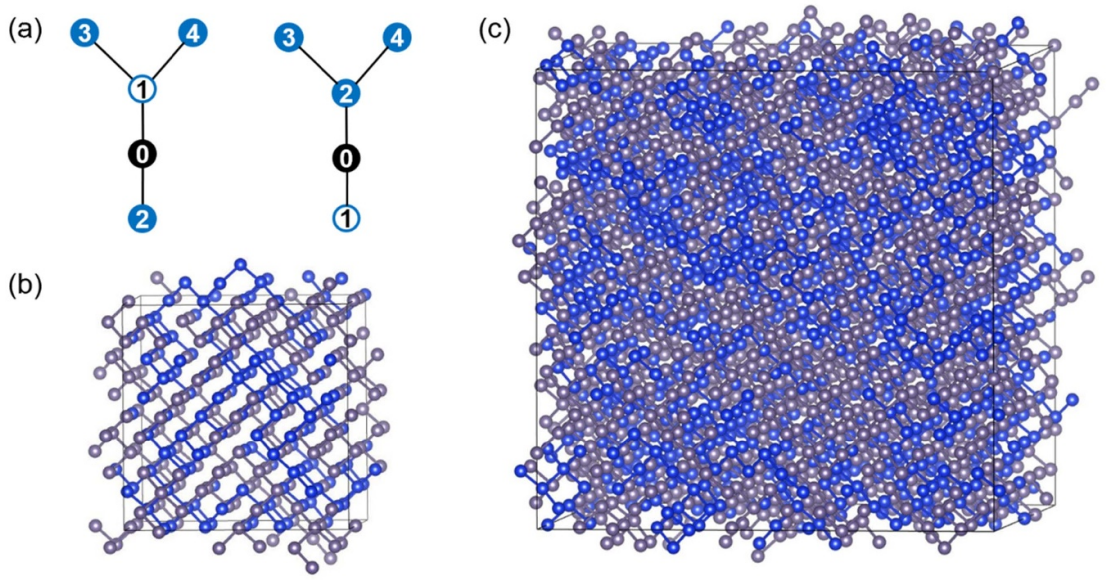

Figure 6. (a) Illustration of the two local structures of atom 0. The empty and full circles denote elemental types A and B, respectively. (b) Snapshot of a 3 × 3 × 3 SiGe random alloy in the training set. (c) Snapshot of a 6 × 6 × 6 SiGe random alloy in the validation set. All the structures are relaxed before calculating the Hamiltonians.

are found. In this work, we collect 66 samples of 2 × 2 × 2 supercells and 1000 samples of 3 × 3 × 3 supercells for training. A snapshot of a 3 × 3 × 3 SiGe random alloy in the training set is shown in figure 6(b). The samples in the validation set are randomly generated with supercell sizes ranging from 2 × 2 × 2 to 6 × 6 × 6. A snapshot of a 6 × 6 × 6 SiGe random alloy in the validation set is shown in figure 6(c). All the structures are relaxed before calculating the Hamiltonians. The structures in the training

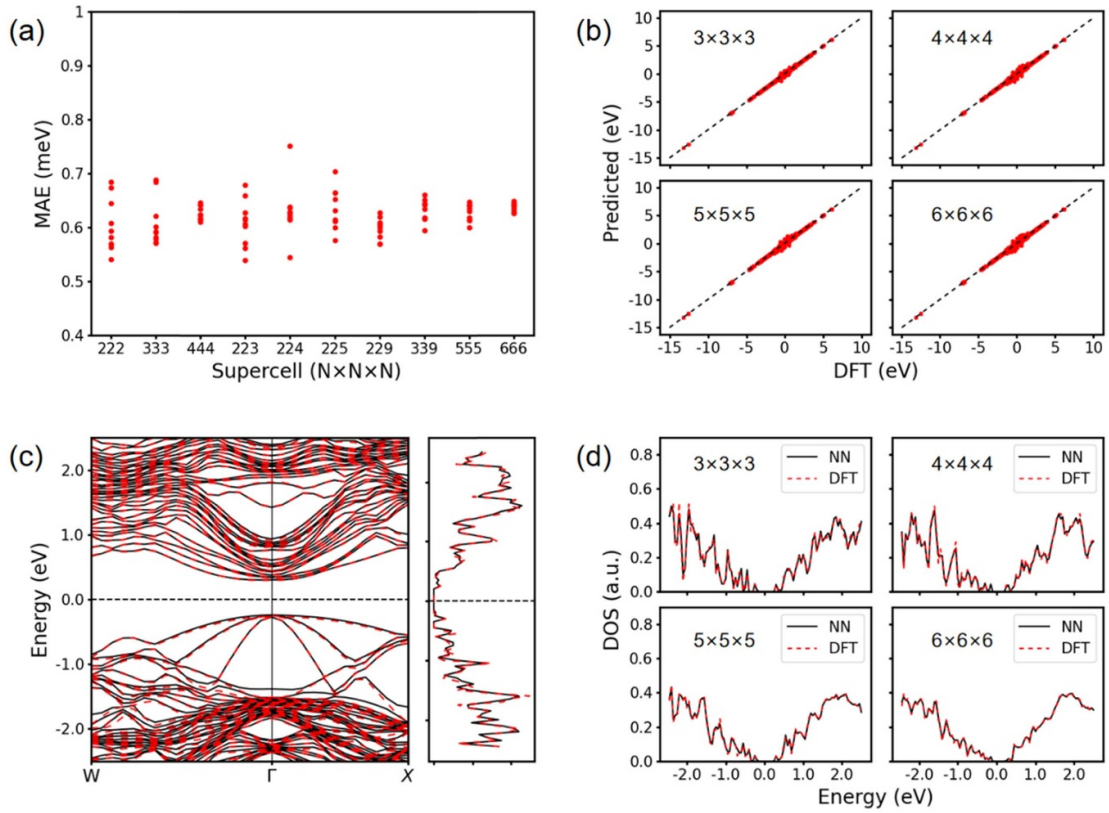

Figure 7. Model evaluation with the SiGe random alloy validation set. (a) The MAE results of hopping parameters. Each point corresponds to an average value of a supercell. (b) Element-wise error distributions of the hopping parameters of $3 \times 3 \times 3$, $4 \times 4 \times 4$, $5 \times 5 \times 5$ and $6 \times 6 \times 6$ supercells. Each point corresponds to an element of the hopping matrix. (c) Band structure and DOS of a $2 \times 2 \times 2$ supercell. (d) DOSs of $3 \times 3 \times 3$, $4 \times 4 \times 4$, $5 \times 5 \times 5$ and $6 \times 6 \times 6$ supercells. Black line: results of graph LC-Net. Red dotted line: DFT results.

set contain up to 216 atoms ($3 \times 3 \times 3$ supercell). For the validation set, the structures contain up to 1728 atoms ($6 \times 6 \times 6$ supercell) and 25 406 784 hopping parameters, demonstrating the extensibility of graph LC-Net. For the training set, the MAEs of the forward model and the backward model are 0.50 meV and 0.52 meV, respectively. For the validation set, the MAEs range from 0.5 to 0.8 meV as shown in figure 7(a). The element-wise error distributions are shown in figure 7(b). The band structures and DOS are calculated using the Hamiltonian predicted by graph LC-Net, and comparisons with DFT calculations are given in figures 7(c) and (d).

3. Discussion

The major advantage of graph LC-Net over DFT calculation is computational efficiency. In DFT calculations, the Kohn–Sham equation needs to be solved iteratively to reach a certain accuracy. For each iteration, eigenvalue problem of a matrix is solved, and the time cost scales cubically. As for the NN model, the iterative calculation is not needed, and the inference time scales linearly with the number of atoms. For the $6 \times 6 \times 6$ supercell with 1728 atoms, graph LC-Net only costs 10.5 s to predict the Hamiltonian. For comparison, the computational time for the DFT self-consistent calculation is about 4800 s using 96 CPU cores, depending on the iteration steps. A comparison regarding the time is summarized in table 1. In this work, graph LC-Net is trained using one GeForce RTX 3090 card, and the DFT calculations are performed using two Intel Xeon Platinum 9242 CPUs (96 cores).

With DFT accuracy and very high computational efficiency, graph LC-Net can be applied to study electronic properties of very large systems, such as high-entropy alloys [28] or defect structures [29]. Graph LC-Net is also expected to accelerate the inverse design of materials with targeted electronic properties [30, 31] such as band gaps, where DFT calculations are performed thousands of times for structure relaxation and property evaluation. The structure relaxations can be efficiently performed using the NN potential models [32], and our graph LC-Net for Hamiltonian can be used to evaluate the electronic properties to avoid the computationally expensive DFT calculations. Note that, our method is not affected by the types of materials directly if only DFT calculation can be performed accurately [33]. Besides the DFT calculation

Table 1. Inference time cost by DFT and graph LC-Net. The size denotes the number of atoms in a structure. The unit of time is in second of CPU time.

Size	64	216	512	1000	1728
DFT	43.2	282.4	1317.2	1271.7	4600.5
Graph LC-Net	0.6	1.3	3.2	6.8	10.5

issue about the training set, the accuracy of the graph LC-Net is possible to improve. For example, the embedding layer could involve more sophisticated initializations [34], and the interaction layers could be replaced by other advanced graph-based models [35].

In conclusion, based on the graph NN architecture and using the local atomic structure information, we have developed the graph LC-Net to predict the hopping parameters and hence obtain DFT Hamiltonian. We have designed the complete LC algorithm for each edge to fix the rotational freedom of the Hamiltonian matrix and reserve the hermiticity of the Hamiltonian to improve the efficiency. By employing graphene and SiGe random alloys as examples, we have demonstrated the high accuracy, extendibility, and efficiency of graph LC-Net in predicting Hamiltonian and electronic properties. This work thus opens doors for studying the electronic properties of large-scale systems using machine learning methods.

4. Methods

4.1. DFT calculation

Before calculating the hopping parameters, all the structures are relaxed into fixed cells using the Vienna *ab initio* Simulation Package [36]. The projector augmented wave [37] type pseudopotential is used and the plane wave energy cut-off is set to 450 eV. The generalized gradient approximation parameterized by Perdew *et al* [38] and the local spin density approximation of Ceperley–Alder [39] are used for the exchange-correlation functional for graphene and SiGe alloy, respectively. The relaxation is stopped when the change of the total energy is smaller than 0.001 eV between two ionic steps.

Then we employ the numerical PAOs as implemented in OpenMX software [40] to perform DFT calculations for the hopping parameters. The PAO is given by a product of a radial function R and a real spherical harmonic function Y by

$$\phi\left(\mathbf{r}\right) = R\left(r\right) Y\left(\theta, \varphi\right), \tag{8}$$

where R is defined numerically and is finite within a cutoff radius. The Hamiltonian and the overlap matrices are given by

$$\begin{aligned} H^{(\mathbf{k})}_{i\alpha,j\beta} &= \sum_n \mathrm{e}^{i\mathbf{R}_n\cdot\mathbf{k}} \left\langle \phi_{i\alpha}\left(\mathbf{r}-\tau_i\right)\middle|\hat{H}\middle|\phi_{j\beta}\left(\mathbf{r}-\tau_j-\mathbf{R}_n\right)\right\rangle, \\ &= \sum_n \mathrm{e}^{i\mathbf{R}_n\cdot\mathbf{k}} h^{(\mathbf{R}_n)}_{i\alpha,j\beta}, \end{aligned} \tag{9}$$

and

$$\begin{aligned} S^{(\mathbf{k})}_{i\alpha,j\beta} &= \sum_n \mathrm{e}^{i\mathbf{R}_n\cdot\mathbf{k}} \left\langle \phi_{i\alpha}\left(\mathbf{r}-\tau_i\right) \middle| \phi_{j\beta}\left(\mathbf{r}-\tau_j-\mathbf{R}_n\right)\right\rangle, \\ &= \sum_n \mathrm{e}^{i\mathbf{R}_n\cdot\mathbf{k}} s^{(\mathbf{R}_n)}_{i\alpha,j\beta}, \end{aligned} \tag{10}$$

where i and α denote atom and orbital, respectively, τ is the atomic position, and $\mathbf{R}_n$ is the lattice vector. According to the principle of locality, the hopping parameters $h^{(\mathbf{R}_n)}_{i\alpha,j\beta}$ are determined by the local structures of atoms i and j within a certain cutoff radius. The overlap parameters $s^{(\mathbf{R}_n)}_{i\alpha,j\beta}$ are calculated by the PAOs without DFT calculations. Then the eigenvalues $\nu_{\mu\mathbf{k}}$ as well as eigenstates $E_{\mu\mathbf{k}}$ of a system are obtained by solving the generalized eigenvalue problem:

$$H^{(\mathbf{k})}\nu_{\mu\mathbf{k}} = E_{\mu\mathbf{k}} S^{(\mathbf{k})}\nu_{\mu\mathbf{k}}. \tag{11}$$

For graphene, the C6.0-s2p2d1 PAOs are used with a cut-off radius of 6.0 Bohr as the basis functions. The Monkhorst–Pack k-meshes of $5 \times 5 \times 1$ are used. For SiGe random alloy, the Si7.0-s2p2d1 and Ge7.0-s2p2d1 PAOs are used as the basis functions and the cut-off radius is set as 7.0 Bohr. For the supercells up to $4 \times 4 \times 4$, the k-points are generated using the automatic scheme to ensure the spacing between k-points is smaller than 0.16 Å^{-1}. For the $5 \times 5 \times 5$ and larger supercells, a single gamma point is used.

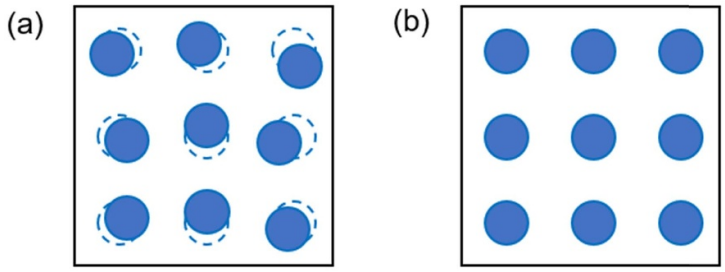

Figure 8. (a) The nearest neighbor is unique (non-degenerate). (b) The nearest neighbor cannot be uniquely determined (degenerate).

4.2. LC

First, consider the case that the nearest neighbor and the second nearest neighbor are different atoms. We use two noncollinear vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ to define the LC as

$$\hat{x}' = \frac{\mathbf{v}_1}{|\mathbf{v}_1|}, \hat{y}' = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|}, \hat{z}' = \hat{x}' \times \hat{y}'. \tag{12}$$

For onsite hopping ($i = j$), the two vectors are determined by

$$\mathbf{v}_1 = \mathbf{r}_{n_1} - \mathbf{r}_i, \mathbf{v}_2 = \mathbf{r}_{n_2} - \mathbf{r}_i, \tag{13}$$

where n_1 and n_2 denote the nearest neighbor and the second nearest neighbor of atom i, respectively.

For offsite hopping ($i \neq j$), we first calculate $d_{\min}(i)$, which is the distance from the nearest neighbor to the target atom i. To utilize the hermiticity, the LCs are defined in two different ways according to the distances. If $d_{\min}(i) < d_{\min}(j)$, we define

$$\mathbf{v}_1 = \mathbf{r}_j - \mathbf{r}_i, \mathbf{v}_2 = \mathbf{r}_{n_{i1}} - \mathbf{r}_i, \tag{14}$$

where n_{i1} denotes the nearest neighbor of atom i. If $d_{\min}(i) > d_{\min}(j)$, we define

$$\mathbf{v}_1 = \mathbf{r}_i - \mathbf{r}_j, \mathbf{v}_2 = \mathbf{r}_{n_{j1}} - \mathbf{r}_j, \tag{15}$$

where n_{j1} denotes the nearest neighbor of atom j. In this way, the h_{ij} and h_{ji} are ensured to be in the same LC. Since the input information for predicting h_{ij} and h_{ji} are the same, the hopping parameters defined by equations (14) and (15) are trained separately. The two models are called the *forward model* and the *backward model*, respectively.

As for the highly symmetric structures where degeneration problems happen, the nearest neighbor and the second nearest neighbor are the same atom, and therefore the above method is not applicable. In this case, we propose algorithm 1 with **Function find_v2()** to determine the basis vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. We show a non-degenerate structure and a degenerate structure in figures 8(a) and (b), respectively. The degeneration problem rarely happens in the graphene dataset as all the atoms are perturbed during the molecular dynamics simulation. For the SiGe dataset, since all the structures are relaxed, the degeneration problem is more likely to arise when the atomic arrangement exhibits some symmetry. Although algorithm 1 contains quite a lot of computational logic, most structures are non-degenerate and will not use algorithm 1. Moreover, algorithm 1 is only used for preprocessing, that is, transforming an atomic structure into a graph. The training and testing process do not involve algorithm 1.

Algorithm 1. Local coordinate

Input: Atomic positions $\{\mathbf{r}_i\}$, neighbors' distances $\{d_{ik}\}$ (sorted)
/* Find two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ to define the LC */
if $i = j$ **then**
  if $d_{i0} < d_{i1} < d_{i2}$ **then** /* the nearest and second nearest neighbors are unique */
    The $\mathbf{v}_1$ and $\mathbf{v}_2$ are determined by the 1st and 2nd neighbors of atom i, respectively
  else
    Get candidates for $\mathbf{v}_1$: $\{$ List $\mathbf{v}_1\}$ and $\mathbf{v}_2$: $\{$ List $\mathbf{v}_2\}$
    if the size of $\{$ List $\mathbf{v}_1\}$ > 1 **then**
      Find the vector with the largest x component as v_1
      If more than one vectors share the same largest x, compare their y and z components in sequence
    end if
    if the size of $\{$ List $\mathbf{v}_2\}$ > 1 **then**
      $\mathbf{v}_2 \leftarrow$find_v2($\{$ List $\mathbf{v}_2\}$, v_1)
    end if
  end if
else
  /* $i \neq j$, v_1 could be either $\mathbf{r}_i - \mathbf{r}_j$ or $\mathbf{r}_j - \mathbf{r}_i$ */
  Calculate $\{$ List $\mathbf{v}_2(i)\}$ and $\{$ List $\mathbf{v}_2(j)\}$[a]
  if size of $\{$ List $\mathbf{v}_2(i)\}$ $= 1$ **then**
    $\mathbf{v}_1 \leftarrow \mathbf{r}_j - \mathbf{r}_i$
else if size of $\{$ List $\mathbf{v}_2(j)\}$ $= 1$ **then**
  $\mathbf{v}_1 \leftarrow \mathbf{r}_i - \mathbf{r}_j$
else
  $\mathbf{v}_2(i) \leftarrow$**find_v2**($\{$ List $\mathbf{v}_2(i)\}$, $\mathbf{r}_j - \mathbf{r}_i$)
  $\mathbf{v}_2(j) \leftarrow$**find_v2**($\{$ List $\mathbf{v}_2(j)\}$, $\mathbf{r}_i - \mathbf{r}_j$)
  $\mathbf{v}_2 \leftarrow \mathbf{v}_2(i)$ or $\mathbf{v}_2(j)$[b]
  end if

[a] $\{$ List $\mathbf{v}_2(j)\}$: candidates for $\mathbf{v}_2$ if $\mathbf{v}_1 = \mathbf{r}_i - \mathbf{r}_j$.
[b] The function **find_v2()** calculates the angle between $\mathbf{v}_2$ and an auxiliary vector $\mathbf{v}_{\mathrm{aux}}$, chose the one with a smaller angle.

Function find_v2($\{$ List $\mathbf{v}_2\}$, $\mathbf{v}_1$)

Inputs:
$\{$ List $\mathbf{v}_2\}$: candidates for $\mathbf{v}_2$
  $\mathbf{v}_{\mathrm{aux}}$: an auxiliary vector[a]
Procedure:
  Calculate the rotation matrix $\mathbf{R}$ to transform v_1 to (1, 0, 0)
  $\alpha \leftarrow \pi$
  for $\mathbf{v}$ in $\{$ List $\mathbf{v}_2\}$ **do**
    Calculate the vector $\mathbf{v}$ in the rotated coordinate
    Calculate the angle α between $\mathbf{v}$ and $\mathbf{v}_{\mathrm{aux}}$
    if a smaller α is found **then**
      $\mathbf{v}_2 \leftarrow \mathbf{v}$
    end if
  end for

[a] The auxiliary vector should be chosen to avoid ambiguities, i.e. there are not two or more vectors in the $\mathbf{v_2}$ list sharing the same α. In this work, $\mathbf{v}_{\mathrm{aux}} = (3.4, 4.5, 5.6)$ is used.

4.3. Rotation of Hamiltonian

The rotation transformation with real spherical harmonics basis is rather complicated [41]. To be convenient, we first convert real spherical harmonics basis into complex spherical harmonics basis, and then the rotation transformation R can be done with the Wigner D-matrix $D^l_{mm'}(R)$, where l is the angular quantum number and m is the magnetic quantum number. The rotation of complex spherical harmonics is given by

$$Y_l^m(\mathbf{r}') = \sum_{m'=-l}^{l} \left[D^l_{mm'}(R)\right]^* Y_l^{m'}(\mathbf{r}). \tag{16}$$

Note that the Wigner D-matrix is unitary, that is, $D_{mm'}(R)^* = D_{m'm}\left(R^{-1}\right)$. The rotation transformation of the hopping parameter matrix $h_{i\alpha,j\beta} \equiv \langle \phi_{i\alpha} | \hat{H} | \phi_{j\beta} \rangle$ is given by

$$h'_{i\alpha,j\beta} = \sum_{a,b} D^{l_\alpha}_{\alpha,a}\left(R^{ij}\right) h_{ia,jb} D^{l_\beta}_{b,\beta}\left(\left(R^{ij}\right)^{-1}\right), \tag{17}$$

where i, j denote atoms, and a, b, α, β denote orbitals. Finally, the hopping parameters are converted back to real spherical harmonics basis.

4.4. Training details

The graph LC-Net model is developed using the Pytorch-Geometric [42] Python library. The variance of the Gaussian radial basis function σ^2 is set to 0.0044 Å^{-2}. The parameters $\beta = 0.25$ and $\gamma = 2.0$ are used in the loss function. Adam optimizer [43] was used with an initial learning rate of 0.001. The reduce-on-plateau strategy is used to schedule the learning rate, and the lower bound of the learning rate is 1×10^{-5}. The number of trainable parameters is 524 731. In this work, the graph LC-Net is trained on a single GeForce RTX 3090 card. The model is trained for 16 h (1336 epochs) on the graphene dataset with 500 samples. The model is trained for 12 days (3752 epochs) on the SiGe dataset with 1066 samples.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/satori555/Graph-LC-Net.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 12188101, 61904035, 11974078, 11991061). Computations are performed at the Supercomputer Center of Fudan University.

Conflict of interest

The authors declare no conflict of interest.

Author contributions

M S, J Y, and X G conceived the idea and designed the research. M S performed the DFT calculations and implemented the codes. X G supervised the work. All authors discussed the results and were involved in the writing of the manuscript.

ORCID iDs

Mao Su https://orcid.org/0000-0003-2066-881X
Hong-Jun Xiang https://orcid.org/0000-0002-9396-3214
Xin-Gao Gong https://orcid.org/0000-0001-7539-5471

References

[1] Jones R O 2015 Density functional theory: its origins, rise to prominence, and future *Rev. Mod. Phys.* **87** 897–923
[2] Xie T and Grossman J C 2018 Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties *Phys. Rev. Lett.* **120** 145301
[3] Chen C, Ye W, Zuo Y, Zheng C and Ong S P 2019 Graph networks as a universal machine learning framework for molecules and crystals *Chem. Mater.* **31** 3564–72
[4] Behler J and Parrinello M 2007 Generalized neural-network representation of high-dimensional potential-energy surfaces *Phys. Rev. Lett.* **98** 146401
[5] Zhang L, Han J, Wang H, Car R and Weinan E 2018 Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics *Phys. Rev. Lett.* **120** 143001
[6] Kang P L, Shang C and Liu Z P 2020 Large-scale atomic simulation via machine learning potentials constructed by global potential energy surface exploration *Acc. Chem. Res.* **53** 2119–29
[7] Prodan E and Kohn W 2005 Nearsightedness of electronic matter *Proc. Natl Acad. Sci.* **102** 11635–8
[8] Kohn W 1996 Density functional and density matrix method scaling linearly with the number of atoms *Phys. Rev. Lett.* **76** 3168–71
[9] Ozaki T 2003 Variationally optimized atomic orbitals for large-scale electronic structures *Phys. Rev.* B **67** 155108
[10] Hegde G and Bowen R C 2017 Machine-learned approximations to density functional theory Hamiltonians *Sci. Rep.* **7** 42669

[11] Schutt K T, Gastegger M, Tkatchenko A, Muller K R and Maurer R J 2019 Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions *Nat. Commun.* **10** 5024
[12] Gu Q, Zhang L and Feng J 2022 Neural network representation of electronic structure from *ab initio* molecular dynamics *Sci. Bull.* **67** 29–37
[13] Li H, Wang Z, Zou N, Ye M, Xu R, Gong X, Duan W and Xu Y 2022 Deep-learning density functional theory Hamiltonian for efficient *ab initio* electronic-structure calculation *Nat. Comput. Sci.* **2** 367–77
[14] Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 Neural message passing for quantum chemistry *Proc. 34th Int. Conf. on Machine Learning* (PMLR) pp 1263–72
[15] Schutt K T, Sauceda H E, Kindermans P J, Tkatchenko A and Muller K R 2018 SchNet—a deep learning architecture for molecules and materials *J. Chem. Phys.* **148** 241722
[16] Choudhary K and DeCost B 2021 Atomistic line graph neural network for improved materials property predictions *npj Comput. Mater.* **7** 1038
[17] Xiong J, Xiong Z, Chen K, Jiang H and Zheng M 2021 Graph neural networks for automated de novo drug design *Drug Discovery Today* **26** 1382–93
[18] Kim Y 2014 Convolutional neural networks for sentence classification (arXiv:1408.5882v2)
[19] Zhang Y and Wallace B C 2016 A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification (arXiv:1510.03820v4)
[20] Brody S, Alon U and Yahav E 2022 How attentive are graph attention networks? (arXiv:2105.14491v3)
[21] Veličković P, Cucurull G, Casanova A, Romero A, Liò P and Bengio Y 2018 Graph attention networks (arXiv:1710.10903v3)
[22] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 770–8
[23] Maas A L, Hannun A Y and Ng A Y 2013 Rectifier nonlinearities improve neural network acoustic models *Proc. ICML* (Citeseer) p 3
[24] Howard A *et al* 2019 Searching for MobileNetV3 (arXiv:1905.02244v5)
[25] Yang Y, Zha K, Chen Y-C, Wang H and Katabi D 2021 Delving into deep imbalanced regression (arXiv:2102.09554v2)
[26] Lin T-Y, Goyal P, Girshick R, He K and Dollár P 2017 Focal loss for dense object detection (arXiv:1708.02002v2)
[27] Wang X and Zhang M 2022 Graph neural network with local frame for molecular potential energy surface (arXiv:2208.00716v1)
[28] Ye Y F, Wang Q, Lu J, Liu C T and Yang Y 2016 High-entropy alloy: challenges and prospects *Mater. Today* **19** 349–62
[29] Wu X, Ming C, Shi J, Wang H, West D, Zhang S and Sun Y-Y 2022 Defects in statically unstable solids: the case for cubic perovskite A-$CsPbI_3$ *Chin. Phys. Lett.* **39** 046101
[30] Chen H-Z, Zhang Y-Y, Gong X and Xiang H 2014 Predicting new TiO_2 phases with low band gaps by a multiobjective global optimization approach *J. Phys. Chem.* C **118** 2333–7
[31] Zhang Y-Y, Gao W, Chen S, Xiang H and Gong X-G 2015 Inverse design of materials by multi-objective differential evolution *Comput. Mater. Sci.* **98** 51–55
[32] Su M, Yang J-H, Liu Z-P and Gong X-G 2022 Exploring large-lattice-mismatched interfaces with neural network potentials: the case of the CdS/CdTe heterostructure *J. Phys. Chem.* C **126** 13366–72
[33] Krukau A V, Vydrov O A, Izmaylov A F and Scuseria G E 2006 Influence of the exchange screening parameter on the performance of screened hybrid functionals *J. Chem. Phys.* **125** 224106
[34] Unke O T, Chmiela S, Gastegger M, Schutt K T, Sauceda H E and Muller K R 2021 SpookyNet: learning force fields with electronic degrees of freedom and nonlocal effects *Nat. Commun.* **12** 7273
[35] Wang Z, Wang C, Zhao S, Xu Y, Hao S, Hsieh C Y, Gu B-L and Duan W 2022 Heterogeneous relational message passing networks for molecular dynamics simulations *npj Comput. Mater.* **8** 53
[36] Kresse G and Furthmüller J 1996 Efficiency of *ab-initio* total energy calculations for metals and semiconductors using a plane-wave basis set *Comput. Mater. Sci.* **6** 15–50
[37] Kresse G and Joubert D 1999 From ultrasoft pseudopotentials to the projector augmented-wave method *Phys. Rev.* B **59** 1758–75
[38] Perdew J P, Burke K and Ernzerhof M 1996 Generalized gradient approximation made simple *Phys. Rev. Lett.* **77** 3865–8
[39] Ceperley D M and Alder B J 1980 Ground state of the electron gas by a stochastic method *Phys. Rev. Lett.* **45** 566–9
[40] Boker S *et al* 2011 Openmx: an open source extended structural equation modeling framework *Psychometrika* **76** 306–17
[41] Dachsel H 2006 Fast and accurate determination of the Wigner rotation matrices in the fast multipole method *J. Chem. Phys.* **124** 144115
[42] Fey M and Lenssen J E 2019 Fast graph representation learning with PyTorch Geometric (arXiv:1903.02428v3)
[43] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980v9)