Learning local equivariant representations for quantum operators prediction

Zhanghao Zhouyin,[1,2,*] Qiangqiang Gu[†],[1,3,†] Zixi Gan,[1] Shishir Kumar Pandey,[1] and Linfeng Zhang[1,4]

[1] *AI for Science Institute, Beijing 100080, China*
[2] *College of Intelligence and Computing, Tianjin University, Tianjin, China, 300350*
[3] *School of Mathematical Science, Peking University, Beijing 100871, China*
[4] *DP Technology, Beijing 100080, China*

(Dated: June 19, 2024)

Simulating electronic behaviour in materials and devices with realistic large system sizes remains a formidable task within the *ab initio* framework. We propose DeePTB, an efficient deep learning-based tight-binding (TB) approach with *ab initio* accuracy to address this issue. By training with *ab initio* eigenvalues, our method can efficiently predict TB Hamiltonians for unseen structures. This capability facilitates efficient simulation of large-size systems under external perturbations like strain, which are vital for semiconductor band gap engineering. Moreover, DeePTB, combined with molecular dynamics, can be used to perform efficient and accurate finite temperature simulations of both atomic and electronic behavior simultaneously. This is demonstrated by computing the temperature-dependent properties of a GaP system with 10^6 atoms.

It has been well-proven that the Density Functional Theory(DFT) has been the perfect backbone for material physics, chemistry and biology. The behaviour of electrons decides most of the properties of matter on Earth. Which is vital to describe phenomena from chemical reactions to electronic transport. Despite the accurate treatment of electrons, the DFT suffers from the self-consistent procedure, which scales cubicly with the system sizes. Meanwhile, the accuracy of DFT is highly decided by the XC functions, where highly accurate methods often describe the electrons. For example, to accurately compute the bandgap of semiconductors, xxx.

A recent practice use machine learning to accelerate the DFT calculation. Some try to reduce the cost of evaluating high-level XC functionals by replacing them with trainable neural networks. Such networks often take in density and output potential, Others however, try to skip the time-consuming self-consistent steps by using neural networks to predict the DFT's output quantum tensors (charge density, overlapping matrix, wave function or Hamiltonian) directly. Such a method requires a localized basis to project the properties to localized features, which then gain transferability by learning the mapping from the atomic environment. Bypassing the self-consistent calculations, such methods have the potential to scale up the electronic structure calculation to structures of millions of atoms.

Early attempts use Gaussian regressions, kernel-based models and Neural Networks to predict the invariant Hamiltonian blocks on localized frames. Recently, a set of highly powerful models of equivariant MPNNs have shown remarkable accuracy. Such networks guarantee the equivariance of output tensor blocks and therefore respect the physical priors of the atomic systems. These models In most practice, initialise the node and edge features with atomic species and the weighted projection of Spherical Harmonics Coefficient, then use a two-body update to build many-body interactions of node and edge features iteratively. Despite achieving high accuracy, such updates would inevitably enlarge the receptive field of node and edge features with many effective neighbours, which limited the parallelization hence the scalability of the model. The storage-demanding nature of quantum tensor prediction tasks is even more severe than this. Since it has very high dimensional features in edge and node, it requires very large GPU memory when doing training and inference. It would be very hard to train such a model on large datasets and finally, evolve to a unified DFT model as MLIPs do [cite big potential models] which need to train on millions of samples. Luckily, this enlarged receptive field is not necessary in many systems where the screening effect takes place. The predicted quantities are usually very local in space, therefore, we can truncate the receptive field while maintaining good accuracy.

[cite allegro] recently proposed a new local updating method of MLIPs to construct a local model with accuracy comparable to E-MPNN. However, unlike modelling potential surfaces one cares about the 0-th order of tensor (energy) and 1-st order (forces) only, the prediction of quantum tensor requires strict supervising on each node/edge features of high-order spherical tensors. This requires the model to have strict locality on both node and edge updates while maintaining sufficient representation capacity to fit the target. Therefore, A vital question would be, how could one use the strictly localized skim to build a powerful equivariant atom and bond features, that can fit and predict accurately and efficiently the corresponding quantum tensor block elements?

Regardless of the localization, another important problem of quantum tensor prediction is the computational complexity. To mix the features of different orders while remaining equivariant, one needs tensor production that scales $O(L^6)$ with the tensor orders. To reach very high accuracy, this complex computation often requires up to

* These authors contributed equally.
† guqq@bjaisi.com; guqq@pku.edu.cn

hundreds of thousands of steps, which significantly limits the power of current models to apply to atom species of larger shells of electrons. This would further prevent the emergence of a unified DFT model that generalizes across the periodic table.

This work is dedicated to presenting an elegant method, the Strictly Localized Equalvariant Massage-passing model (named Slem), which uses a fully localized skim to build high-order node and edge equivalent features. The model embedded localized edge hidden states, and used them to construct localized node and edge features without including distant neighbours beyond the prefixed cutoff range. This allows the Slem model to generalize better, and scale to larger systems. To achieve accuracy, a fast but efficient SO(2) convolution is implemented, with edge-specific training weights that process each edge separately. Benefiting from this, our method receive very high accuracy, which is even better than the nonlocal method reported. Meanwhile, we also provide a non-local version named Lem for the user when the locality breaks.

a. Message-passing Neural Networks The message-passing neural networks have been widely applied in modelling atomic systems, which have perfect accuracy in capturing delicate relations between atomic environment and physical properties. In the Message-passing scheme, the atoms are treated as nodes in graphs, whose bonds with the neighbouring atoms are treated as connected edges within a certain cutoff radius. The embedded atomic features then, will be processed by trainable operation functions, generating messages from each edge to update the embedding of central atoms. Formally, the framework of MPNN can be summarized as:

$$\begin{aligned}
\mathbf{e}^{ij,L} &= \mathcal{N}_L\left(\mathbf{n}^{i,L-1}, \mathbf{n}^{j,L-1}, \mathbf{e}^{ij,L-1}\right) \\
\mathbf{m}^{ij,L} &= M_L\left(\mathbf{n}^{i,L-1}, \mathbf{n}^{j,L-1}, \mathbf{e}^{ij,L}\right) \\
\mathbf{n}^{i,L} &= U_L\left(\mathbf{n}^{i,L-1}, \sum_{j\in\mathcal{N}(i)} \mathbf{m}^{ij,L}\right)
\end{aligned}$$

Benefiting from this updating framework, many-body interactions and long-term dependence would be constructed. This enables good performance on various applications. Previous work was mostly developed based on such updating scheme, which has achieved remarkably high precision in reported systems. However, as the updates go on, the features of each atom would be communicated with an effective cutoff radius, that grows linearly as the update steps. Therefore, the final output features would have an effective neighbour list that scales cubicly as the effective cutoff radius, which makes parallelization intractable. This is particularly important in fitting quantum tensors since the cutoff radius that aligns with the cutoff of the LCAO basis is often larger than other tasks.

A localized attempt for such updates that constructs many-body interactions without including distant atoms is proposed. The allegro model, for example, changes the

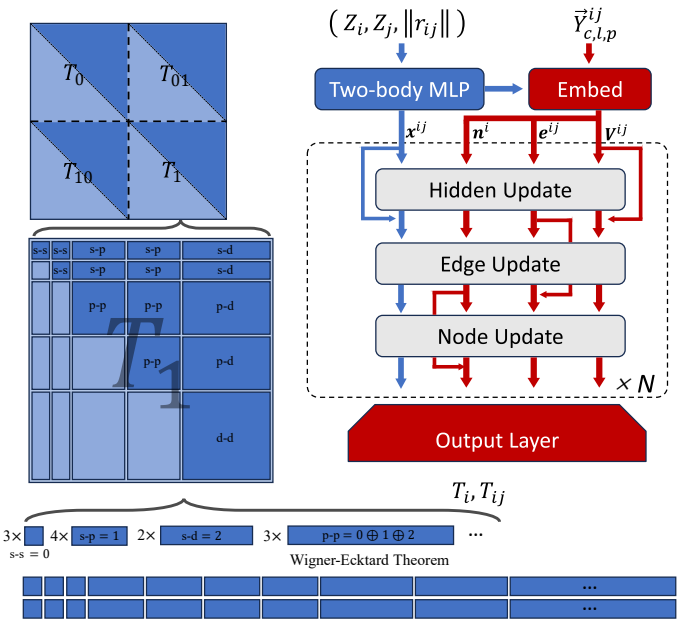

FIG. 1.

updating rules by incorporating a hidden atom-pair state, which is explicitly dependent on the direction vector and the centre atom embedding only. This framework works successfully in potential energy prediction while requiring further improvement when fitting more general edge and node features. other methods ... .

b. Equivariant Message Passing There are certain natures of the physical quantities that should be invariant or equivariant under e.g. the spatial and temporal operations. For example, under translation, rotation and reflection, the energy of a system is invariant, while the forces on each atom should transform accordingly. Such three kinds of operations constitute the group operations of E(3), known as the Euclidian Group (O(3) if without translation). To model such equivariant quantities, a set of neural network models, which consists of operations that preserve such equivariance is developed. This kind of neural network possesses physical priors since they are composed of operations that ensure the output is transformed simultaneously as the transform of input, therefore appears more generalisable, accurate and data-efficient in predicting physical quantities. Formally an equivariant operation on vector space X to vector space Y is such that:

$$f(D_X[g]x) = D_Y[g]f(x) \quad \forall g \in G, \forall x \in X$$

Where $D_X[g] \in GL(X)$ is the representation of group element g on vector space X. Since the representation takes the form of the direct sum of irreducible representations of the group G, it suggests that the vectors x and y themselves are composed by the irreducible representation (irreps for short) of group G. Usually, we consider here for operations of group O(3)(since translational symmetry is naturally satisfied by using relative coordinates),

the irreps are the spherical tensors with index l and p. So features x, y should also be indexed by l, p. For each irrep with degree l, an internal degree of freedom, denoted by m exists such that $|m| \leq l$. Therefore, each irreps of order l belong to a subspace of dimension $2l+1$. Once the l, p index is defined, one can construct the group operations, which is Wigner-D matrix of degree l, shaped $(2l+1) \times (2l+1)$.

Doing numerical operations on such irreps features following certain rules. For the tensors with same degree l, addition and subtraction can be conducted. To mixed up the tensors of different l index, an important operation called tensor product is applied, which is considered as the generalized operation of multiplication on spherical tensors. More formally:

$$(\mathbf{x} \otimes \mathbf{y})^{l_3}_{m_3} = \sum_{m_1,m_2} C^{(l_3,m_3)}_{(l_1,m_1)(l_2,m_2)} \mathbf{x}^{l_1}_{m_1} \mathbf{y}^{l_2}_{m_2}$$

Where $C^{(l_3,m_3)}_{(l_1,m_2)(l_2,m_2)}$ are Clebsch-Gordan Coefficients. This operation is found extremely useful in constructing edge and node updating functions. For example in []. However, the conventional tensor product has the time and memory scales of $O(l^6_{max})$ where l_{max} is the maximum spherical order l of tensor features of $\mathbf{x}$ and $\mathbf{y}$. Such complexity limits the application like the equivariant forces fields where high inference speed is vital. It threatens quantum tensor prediction tasks even more severely since it requires conducting tensor products on very high-order spherical tensors. For example, constructing a block of $d-d$ orbital pair requires irreps of maximum order $l=5$, and $l=7$ for $f-f$ block. Such high costs make the training impossible.

c. Parameterize Equivariant Quantum Tensors The quantum tensors such as Hamiltonian, Overlaps and Density Matrix in Density Functional Theory follow the equivariance of $E(3)$ group. Therefore, can be expressed in tensorized features using group theory. The procedure diagrammed is illustrated in 1. Specifically, $E(3)$ group consist of spatial rotation, translation, and reflection. The translational symmetry is automatically satisfied by performing operations only on relative coordinates. Here, rotation and reflection need special treatments. Under LCAO basis, the quantum tensor block can be expressed as:

$$T^{i,j}_{l_1,l_2,m_1,m_2} = \langle i, l_1, m_1 | \hat{T} | j, l_2, m_2 \rangle$$

Here i, j stands for atom site i and j, and l_1, and l_2 denotes the atomic orbital of site i's l_1 components and site j's l_2 components. $T^{i,j}_{l_1,l_2,m_1,m_2}$ are the matrix element. Assume we map the coordinate of the current system by a rotation operation R, the equivariance requires the quantum tensor transform accordingly as:

$$T'^{i,j}_{l_1,l_2,m'_1,m'_2} = \sum_{m_1,m_2} D^{l_1}_{m_1,m'_1}(\mathbf{R}) D^{l_2}_{m_2,m'_2}(\mathbf{R})^* T^{i,j}_{l_1,l_2,m_1,m_2}$$

Where $D^l(R)_{m,m'}$ is the Wigner-D matrix of order l. To construct such equivariance blocks from irreps, the Wigner-Eckart Theorem can be applied, which decomposes the operator indexed by l_1, l_2 into a single spherical index l_3 that satisfied $|l_1 - l_2| \leq l_3 \leq (l_1 + l_2)$. Which states:

$$t^{i,j}_{l_3,m_3} = \sum_{l_1,m_1,l_2,m_2} C^{(l_3,m_3)}_{(l_1,m_1)(l_2,m_2)} T^{i,j}_{l_1,l_2,m_1,m_2}$$

In this manner, for onsite ($i=j$) and hopping ($i \neq j$), features $t^{i,j}_{c,l,m}$ corresponding directly to the edge and node features of the atomic graph can be computed (extra c for there would exist multiple features for one spherical order). It is worth mentioning that by using the Hermitian nature of quantum tensors, only the elements of upper diagonal blocks need to be parameterized since the remaining elements can be transformed from these by simply using matrix transpose. What we need to do is to generate the node and edge features, and supervise them on such decomposed target features. Once certain accuracy is attained, we can use the inversion of Wigner Ecktard Theorem to reconstruct the predicted Hamiltonian blocks, as:

$$T^{i,j}_{l_1,l_2,m_1,m_2} = \sum_{l_3,m_3} C^{(l_3,m_3)}_{(l_1,m_1)(l_2,m_2)} t^{i,j}_{l_3,m_3}$$

The whole procedure satisfies the rotational and transformational symmetry. Therefore achieves the equivariance requirement. One last operation that needs consideration is the inversion. This is important in periodic systems since hopping between primitive lattice cells appears often. However, this equivariance can be simply achieved by assigning different parity to the features t.

This paper is arranged as below. In the Results section, the design of the model Slem will be illustrated in detail. Then we compared the accuracy of our model on reported datasets with reported models. Later, the generalization experiment is conducted.

Results

This section describes the methods to learn local equivariant many-body interactive features for DFT quantum tensor predictions.

A. Rescale the node and edge features

As the first step, we normalise the features that the models learn via a rescaling operation. The target node/edge equivariant features $h^i_{c,l,p}$ and $h^{ij}_{c,l,p}$ can be decomposed into constant norms and biases, and the normalized features that have balanced variance. Formally:

$$\mathbf{t}^i_{c,l,p} = \sigma^{Z_i}_{c,l,p} \hat{\mathbf{t}}^i_{c,l,p} + \mu^{Z_i}_{c,l,p} \delta_l$$
$$\mathbf{t}^{ij}_{c,l,p} = \sigma^{Z_i,Z_j}_{c,l,p} \hat{\mathbf{t}}^{ij}_{c,l,p} + \mu^{Z_i,Z_j}_{c,l,p} \delta_l$$

Where Z_i, Z_j are the atom species of atom i and j, $\sigma^{Z_i}_{c,l,p}$, $\sigma^{Z_i,Z_j}_{c,l,p}$, $\mu^{Z_i}_{c,l,p}$ and $\mu^{Z_i,Z_j}_{c,l,p}$ are norm and bias value for each atom and atom-pair. The values of the stds and biases are counted from the datasets, and then used as weights and shift biases in an atom/bond type-specific scaling layer.

The help to conduct this is evident. For the tensor operators in LCAO basis, the diagonal blocks (or node features) often have very unbalanced norms across irreps of different l, which prevent the model from learning a smooth function to describe. On the other hand, different irreps in the off-diagonal blocks (or edge features) have diversified decaying behaviour, it would be hard to learn such decaying implicitly. Instead, normalizing in this way makes the decay function for each irreps belong to a value ranging from 0 to 1, it would be very easy to use hidden states and a few layers of MLP activated by ReLU-like function to project the decaying value explicitly from the preserved hidden scalar features for each edge. The analysis of the decaying behaviour of edge irreps and the norm of node irreps can be found in the supplementary.

B. The Slem model

The Slem model architecture is displayed in 1. Generally, the model maintains a set of features, including hidden features $\mathbf{x}^{ij,L}$, $\mathbf{V}^{ij,L}_{c,l,p}$, node features $\mathbf{n}^{i,L}_{c,l,p}$ and edge features $\mathbf{e}^{ij,L}_{c,l,p}$. The hidden features specifically, contain a scalar channel and a tensor channel, which interact with each other, creating an ordered atom-pair representation to construct local node and edge representations. After iterative updates, the representation will be scaled by the statistical norm and biases, therefore attaining the final prediction.

Feature initialization Firstly, the initial scalar hidden feature is computed from the two-body embeddings of atom species Z_i and Z_j, and the radial distance, by:

$$\mathbf{x}^{ij,L=0} = \mathbf{MLP}_{\text{two-body}}\left(\mathbf{1Hot}(Z_i)||\mathbf{1Hot}(Z_j)||\mathbf{B}(r_{ij})\right)\cdot u(r_{ij})$$

The atom species are embedded with one-hot vectors, and a set of trainable Bessel basis $\mathbf{B}(r_{ij})$ is used for distance. $u(r_ij)$ are envelope functions [cite] to add explicit radial dependence. Then, the edge and hidden features are initialized as weighted spherical harmonics of relative edge vectors:

$$\begin{aligned}\mathbf{V}^{ij,L=0}_{c,l,p} &= w_{c,l,p}\left(LN\left(\mathbf{x}^{ij,L=0}\right)\right)\vec{Y}^{ij}_{l,p}\\ \mathbf{e}^{ij,L=0}_{c,l,p} &= w_{c,l,p}\left(\mathbf{x}^{ij,L=0}\right)\vec{Y}^{ij}_{l,p}\end{aligned}$$

Here the weights are learnt from the initialized scalar hidden features $\mathbf{x}^{ij,L=0}$. We use layer-norm LN to make sure the hidden tensor features have a balanced amplitude of each edge.

The initial node features then, are linear transformations of the aggregated edge features, which is:

$$\mathbf{n}^{i,L=0}_{c,l,p} = \mathbf{Linear}\left(\frac{1}{\sqrt{N_{avg}}}\sum_{j\in N(i)}\mathbf{e}^{ij,L=0}_{c,l,p}\right)$$

1. Speed up tensor product

Before proceeding to the updating of model Leven, we need to review the tensor production operation. To fuse the information of the equivariant features, we use tensor product in all the updating blocks. Generally, the tensor product in the Leven model is performed with the concatenated equivariant features $\hat{\mathbf{f}}^{ij}_{c_3,l_3}$ and the weighted projection of the edge shift vector on the spherical harmonics function $\vec{Y}^{ij}_{l_2}$. Formally:

$$\begin{aligned}\mathbf{f}^{ij}_{c_3,l_3} &= \hat{\mathbf{f}}^{ij}_{c_1,l_1}\otimes w^{ij}_{c_2,l_2}\vec{Y}^{ij}_{l_2}\\ &= \sum_{c_1,l_1,l_2}\hat{w}^{ij}_{c_1,l_1,l_2}\sum_{m_1,m_2}C^{(l_3,m_3)}_{(l_1,m_1)(l_2,m_2)}\\ &\quad\cdot f^{ij}_{c_1,l_1,m_1}Y^{ij}_{l_2,m_2}\end{aligned} \tag{1}$$

Here $\hat{w}^{ij}_{c_1,l_1,l_2} = \sum_{c_2} w_{c_1,c_2,l_1,l_2}w^{ij}_{c_2,l_2}$ are edge specific parameters for each tensor product operation. Performing such tensor products on features of very high orders is extremely cumbersome. Therefore, we applied the recently developed SO(2) convolution to simplify, reducing the computation and storage complexity from $O(l^6_{max})$ to $O(l^3_{max})$.

The simplification idea is intuitive. $\vec{Y}^{ij}_{l_2,m_2}$ are sparse tensors if rotated to align with the edge ij, which is nonzero only for $m_2 = 0$. Therefore, it is easier to compute the tensor production in the direction of edge ij, and rotate inversely the output afterwards. This step remove the m_2 index in summation of Eq. 1. Besides, when we considering CG-coefficients with $m_2 = 0$, it turns out $C^{(l_3,m_3)}_{(l_1,m_1)(l_2,0)} = 0$ except for $m_3 = \pm m_1$. In this case, we can further reduce the summation in Eq. 1 by replacing $\pm m_1$ with a single index m. Simplification using these results, the operations can be reformulated formally as:

$$\begin{pmatrix}\mathbf{f}^{ij}_{c,l,m}\\ \mathbf{f}^{ij}_{c,l,-m}\end{pmatrix} = \sum_{c',l'}\begin{pmatrix}\mathbf{w}_{c,c',l',m} & -\mathbf{w}_{c,c',l',-m}\\ \mathbf{w}_{c,c',l',-m} & \mathbf{w}_{c,c',l',m}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{f}^{ij}_{c',l',m}\\ \mathbf{f}^{ij}_{c',l',-m}\end{pmatrix}$$

This is just a linear operation on $\hat{\mathbf{f}}^{ij}_{c',l',m'}$. In this manner, computing very high-order tensor products to 8,9 and 10 can be performed with acceptable costs, which is essential for [xxx type systems where f, or even higher orbital exists]. One last step, we cooperate edge-dependent weights into the new SO(2) tensor product method, by parameterizing the weights as $\hat{\mathbf{w}}^{ij}_{c,c',l',m} = \mathbf{w}_{c,c',l',m}\mathbf{w}^{ij}_{c',l'}$, where $\mathbf{w}^{ij}_{c',l'}$ are mapped by an MLP from hidden scalar

features $\mathbf{x}^{ij,L}$. Equipped with this powerful and efficient tensor product layer, we can construct local interactive updates of the features.

2. Hidden updates

The creation of hidden states is vital to achieving local node and edge equivariant features. Different from conventional edge-like features in MPNNs, the hidden states $\mathbf{x}^{ij}$ and $\mathbf{V}^{ij}_{c,l,p}$ in our method dependent solely on the local environment of centre atom i, the shift vector of edge ij, and the initial information of atom j (i.e. the atom species Z_j). Clearly, such design exclude of neighbours of j into hidden states.

To construct many-body interactions, as in Fig. 1(b), the node features $\mathbf{n}_i$ and hidden tensor features $\mathbf{V}^{ij}_{c,l,p}$ would be concatenated and doing tensor product with the projection coefficients of edge shift vector $\vec{r}_{ij}$ on the spherical harmonics functions. The operation is written formally as:

$$\hat{\mathbf{V}}^{ij,L}_{c_3,l_3,p_3} = \left(\mathbf{n}^{i,L-1}||\mathbf{V}^{ij,L-1}\right)_{c_1,l_1,p_1} \otimes w^{ij,L}_{l_2,p_2} \vec{Y}^{ij}_{l_2,p_2} \quad (2)$$

After the tensor production, the output features will be passed through the gated non-linearity [cite], and transformed by an E3linear layer to mix up the information across different channels. The new hidden feature will be multiplied by the weights learned from normalized scalar features, to include the radial information explicitly.

The scalar hidden features are updated by mixing the 0th order information from $\mathbf{V}^{ij,L}_{c,l,p}$ with a latent MLP, which is:

$$\mathbf{x}^{ij,L} = \mathbf{MLP}\left(\mathbf{x}^{ij,L-1}||\mathbf{V}^{ij,L}_{c,l=0,p=1}\right) \cdot u(r_{ij}) \quad (3)$$

Containing the explicit decaying envelope function $u(r_{ij})$, and the many-body interactions of scalar and tensor features, the sclar hiddien states $\mathbf{x}^{ij,L}$ can be very useful to describing the explicit decaying behaviour of each edge irreps feature along the radial distance.

3. Node updates

The strict local node representation $\mathbf{n}^{ij,L}_{c,l,p}$ can be constructed naturally from the many-body interactive tensor features $\mathbf{V}^{ij,L}_{c,l,p}$. We follows the MPNN style to create the message from node-j to node-i. Formally:

$$\mathbf{m}^{ij,L}_{c_3,l_3,p_3} = \left(\mathbf{n}^{i,L-1}||\mathbf{V}^{ij,L}\right)_{c_1,l_1,p_1} \otimes w^{ij,L}_{l_2,p_2} \vec{Y}^{ij}_{l_2,p_2} \quad (4)$$

Here again, we exclude the neighbouring information of atom-j in $\mathbf{m}^{ij,L}_{c_3,l_3,p_3}$, thanks to the partial dependent updates of $\mathbf{V}^{ij,L}_{c,l,p}$. However, all necessary interactions are contained in this format, including the interactions of atom-j, edge $i-j$, atom i, and the environment of i.

TABLE I. Comparison of MAE in meV for predicting Hamiltonian matrices using Slem and other methods on materials with DFT-LCAO basis up to d, f, and g orbitals. The number in parentheses indicates the number of parameters.

Systems with LCAO-basis up to d-orbitals					
Material	**DeePTB-E3**		**DeepH-E3**		HamGNN
	(0.7M)	(4.5M)	(1.0 M)	(4.5M)	(10M)
MoS_2	**0.34**	**0.14**	0.46	0.55	0.63
Graphene	**0.26**	**0.14**	0.40	0.28	0.40
Si(300K)	**0.10**	**0.07**	0.16	0.10	9.0
Si(600K)	**0.20**	**0.13**	0.19	0.14	9.7
Systems with LCAO-basis up to f and g-orbitals					
	(1.7M)			(1.9M)	
GaN	**0.21**			0.87	
HfO_2	**0.28**			-	

TABLE II. MAE in meV for predicting Density Matrix using Slem on materials with DFT-LCAO basis up to d, f, and g orbitals. The model sizes align with the model in table 1

Slem charge density model				
Materials	Silicon	GaN	HfO_2	$C_7H_{10}O_2$
MAE	3.2e-5	2.3e-5	3.7e-5	-

Each message then is passed through a gated activation and E3Linear layer, and weighted separately by weights learnt from the hidden scalar features, by:

$$\hat{\mathbf{n}}^{ij,L}_{c_3,l_3,p_3} = \frac{1}{N_{avg}} \sum_j \mathbf{w}^{ij}_{c_3,l_3,p_3} \mathbf{m}^{ij,L}_{c_3,l_3,p_3} \quad (5)$$

The central difference of the weighting scheme here and one in hidden updates is that, the weights here (and in the edge updates) are learnt from $\mathbf{x}^{ij,L}$ directly, while the wrights in hidden updates are mapped from normalized $\mathbf{x}^{ij,L}$. Therefore, the absolute radial decay is preserved in the weights, giving strong priors that the message from a closer atom are often more important, while also maintaining the flexibility. Meanwhile, combining the updates of hidden scalar $\mathbf{x}^{ij,L}$, we can see the weights are dependent on the features $\mathbf{V}^{ij,L}$ and $\mathbf{n}^{i,L-1}$. Such style align surprisingly with the graph attention that shows powerful expressiblity in various tasks. Here $\mathbf{w}^{ij}_{c_3,l_3,p_3}$ corresponding to an attention score computed from $\mathbf{V}^{ij,L}$ and $\mathbf{n}^{i,L-1}$.

4. Edge updates

The locality of edge features $\mathbf{e}^{ij,L}_{c,l,p}$ can be naturally preserved using the localized node features. By mixing

TABLE III. Comparison of data-efficiency of Slem model and others using a random splited datasets.

MoS2					
Partition	100%	80%	60%	40%	20%
Slem	**0.34**	**0.37**	**0.39**	**0.37**	**0.37**
DeePH-E3	0.46	0.72	0.84	1.03	1.46
HamGNN	1.11	1.15	1.38	1.65	1.68
Graphene					
Partition	100%	80%	60%	40%	20%
Slem	**0.26**	**0.26**	**0.27**	**0.21**	**0.26**
DeePH-E3	0.40	0.30	0.33	0.36	0.60
HamGNN	0.41		0.37	0.42	0.60

the information of node features on both sides, localized updates can be formulated as:

$$\hat{\mathbf{e}}^{ij,L}_{c_3,l_3,p_3} = \left(\mathbf{n}^{i,L}||\mathbf{V}^{ij,L}||\mathbf{n}^{j,L}\right)_{c_1,l_1,p_1} \otimes w^{ij,L}_{l_2,p_2}\vec{Y}^{ij}_{l_2,p_2}$$

Similarly, the updated edge features are processed via a gated activation and an E3Linear layer, and then multiplied with weights learnt from the hidden scalar features (without normalization) as:

$$\mathbf{e}^{ij,L}_{c_3,l_3,p_3} = \mathbf{w}^{ij}_{c_3,l_3,p_3}\hat{\mathbf{e}}^{ij,L}_{c_3,l_3,p_3}$$

Here we display the updating rules of our model in the MPNN framework, which looks like:

$$\begin{aligned}
\mathbf{V}^{ij,L} &= \mathcal{N}_L\left(\mathbf{n}^{i,L-1}, \mathbf{V}^{ij,L-1}\right) \\
\mathbf{e}^{ij,L} &= \mathcal{V}_L\left(\mathbf{n}^{i,L-1}, \mathbf{V}^{ij,L-1}, \mathbf{n}^{j,L-1}\right) \\
\mathbf{m}^{ij,L} &= M_L\left(\mathbf{n}^{i,L-1}, \mathbf{V}^{ij,L}\right) \\
\mathbf{n}^{i,L} &= U_L\left(\mathbf{n}^{i,L-1}, \sum_{j\in\mathcal{N}(i)} \mathbf{m}^{ij,L}\right)
\end{aligned}$$

This update scheme constructs many-body interactions to build equivariant edge and node features while preserving the absolute locality by excluding atoms outside the constant cutoff radius. Such a scheme, in principle, would have much better transferability, data efficiency, as well as scalability. Most importantly, it is possible to parallelize such update computations, while allowing for training and inference on multiple devices, providing the possibility to inference on very large atomic structures. The following chapter will focus on validating the effectiveness of this framework via learning equivariant DFT Hamiltonians and density matrices, which have very high-order tensors and are sensitive to errors, which need extremely high accuracy for accurate prediction. We demonstrate our method is much better than the previously reported ones in training and inference speed, transferability and the potential power to parralizing inference on multi-GPU devices.

C. Benchmark the accuracy and flexibility

We benchmark the performance of our model on fitting Hamiltonian and Density Matrix operators using a diversified dataset. The datasets contain 2-dimensional systems such as single-layer MoS2 and Graphene, and the 3-dimensional bulk silicon. Moreover, Slem model is particularly powerful for high orders of spherical tensors. Therefore, we also train Slem on bulk GaN and HfO2 systems, whose basis sets include f and g orbitals respectively. The structures in the above systems are sampled with molecular dynamics, where the force and energy are computed via DFT(for MoS2 and Graphene) and trained Neural Network Potentials(other systems). As displayed in Table.I, our Slem model obtains state-of-the-art accuracy in predicting Hamiltonian operators, with the lowest mean absolute error (MAE) in all testing systems compared with the referenced methods. Surprisingly, we find that a comparably small model is sufficient to reach very high accuracy. The band structure of some typical structures computed directly from the Slam predicted Hamiltonian are displayed in [], where the band energies are indistinguishable from the ones from the DFT.

For density matrix operators, the fitting results are displayed in Table.II. The table shows very high accuracy that is even close to the machine accuracy limit of float32 digital numbers. In fig We use the trained model to predict the density matrix, and then transform it to density distribution over real spaces for visualization. This could be particularly important to applications of electron transfer via methods like Differential Charge Density.

Moreover, benefiting from the strict localization scheme, our Slem model is more data efficient. That is to say, we need a smaller dataset to train the model. Here we design an experiment to quantitize the data efficiency performance, by randomly splitting the dataset used above into subsets. Each of the subsets contains 20, 40, 60 and 80 percent of original training data. We then train the strictly local Slem model and other methods with the subsets separately and validate their performance on the original validation sets. The results are displayed in Table.III. We can see the remarkable performance of our method in terms of data efficiency. This allows the user to generate a training set with lower costs. Moreover, the excellent transferability makes it more suitable for being a backbone model towards generating a universal model across the elements in the whole periodic table.

D. Efficiency and Scalability

Many interesting and important properties in material science, chemistry and biology can be analysed with systems composed of heavy atoms. The inclusion of heavy atoms would introduce very high-order spherical tensors when representing the quantum operators such as DFT

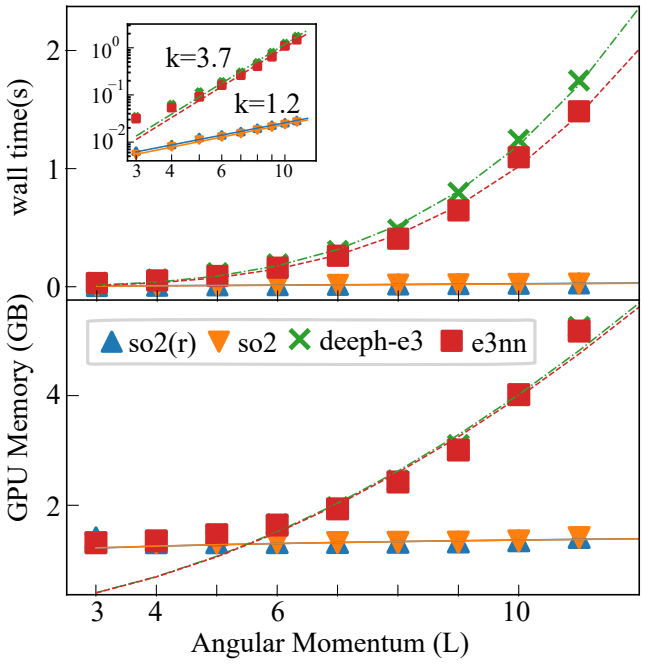

FIG. 2.

Hamiltonian. Therefore, scaling to such systems would be quite a challenge because the tensor product used to construct complex spherical tensors scales as $O(L^6)$. It is very hard to train and inference on systems with heavy atoms using conventional tensor production.

Moreover, one particularly useful aspect of the model is to infer large material systems while training with small structures. Therefore, to scale up to larger systems, it is beneficial to parallelize the computation by assigning partitions of the whole atomic structure to multiple GPU workers. However, this is very difficult for most of the current available equivariant tensor prediction models. Since the increasing receptive fields from the iterative updates of the graph would increase the minimal size of each partitioned subgraph. Such problems are even more severe in tasks such as DFT Hamiltonian and density matrix prediction, which require gradient computation that propagates along with the atomic graph. The Slem model is proposed as a resolve to these problems, which can construct flexible high-order tensor products efficiently while assisting the parallelization via its strict locality.

Firstly, for efficiency, the implementation of SO(2) convolution reduces the tensor product computational complexity to $O(L^3)$, which is then further reduced by the parallelization of matrix operations built-in PyTorch. In Figure.2, we compare the wall time and GPU memory consumed by tensor product operations using SO(2) convolution and the normal method employed in DeePH-E3 and E3NN. Clearly, the tensor production via SO(2) convolution is far more efficient than conventional methods. This reduction of cost makes our method very efficient, which is capable of handling all possible basis choices in LCAO DFT.

[results of experiments that show the efficiency of our model]

Moreover, the strict local design assists the parallelization greatly. Since the edge and node features in Leven model are constructed locality, each node and edge features only dependent on the neighbour subgraph of the whole system, as:

$$\begin{aligned}\mathbf{n}^{i,L} &= U_L\left(\mathbf{n}^{i,L-1}, \sum_{j\in\mathcal{N}(i)} \mathbf{m}^{i,L}\right) \\ &= U(Z_i, \{\vec{Y}^{ij}, Z_j\}_{j\in N(i)})\end{aligned} \tag{6}$$

Therefore, it is possible to divide the whole system into sub-blocks and compute their node and edge features independently on different devices. This is extremely important when expanding the simulation power of Density Functional Theory to very large systems. In fig, we can see for silicon, a typical model can only predict around 10000 atom structures on devices with 32GB memory. Despite the linear dependency, the inference on a system with 10^5 10^7 would require a memory with potentially over 300 GB. This is not available with a currently single card, thus parallelization with multiple GPUs is vital. Therefore, a strictly localized model such as Slem would be a great help to extend the system sizes.

[results of the scalability of the model]

E. A practical demo

Discussion