Molecular Graph Transformer: Stepping Beyond ALIGNN Into Long-Range Interactions

Marco Anselmi, Greg Slabaugh, Rachel Crespo-Otero and Devis Di Tommaso

Graph Neural Networks (GNNs) have revolutionized material property prediction by learning directly from the structural information of molecules and materials. However, conventional GNN models rely solely on local atomic interactions, such as bond lengths and angles, neglecting crucial long-range electrostatic forces that affect certain properties. To address this, we introduce the Molecular Graph Transformer (MGT), a novel GNN architecture that combines local attention mechanisms with message passing on both bond graphs and their line graphs, explicitly capturing long-range interactions. Benchmarking on MatBench and Quantum MOF (QMOF) datasets demonstrates that MGT’s improved understanding of electrostatic interactions significantly enhances the prediction accuracy of properties like exfoliation energy and refractive index, while maintaining state-of-the-art performance on all other properties. This breakthrough paves the way for the development of highly accurate and efficient materials design tools across diverse applications. Code is available at: https://github.com/MolecularGraphTransformer/MGT

1 Introduction

Across various scientific disciplines, from computer vision to chemistry, graphs serve as powerful models for representing systems of objects and interactions. A graph \( G = (V,E) \) consists of a set of nodes \( V \) and a set of edges \( E \) between pairs of nodes, representing the relationship between them. Geometric Deep Learning (GDL) leverages the expressive power of graphs to analyze these systems, providing insights into their underlying structure. Common applications of GDL include shape analysis and pose recognition in computer vision, link and community detection on social media networks, representation learning on textual graphs, and medical image analysis for disease detection and property prediction for molecular and crystalline materials.

In the field of quantum chemistry, the development of Graph Neural Networks (GNN) has provided a means of computing the properties of molecules and solids, without the need to approximate the solution to the Schrödinger equation. Furthermore, compared to other Machine Learning (ML) techniques, they have shown immense potential in the field of chemistry, since they do not require manual feature engineering and have significantly better performance compared to other ML models. GNN models represent molecules or crystalline materials as graphs with a node for each constituent atom and an edge as bonds or interatomic relationships. By passing messages through the edges they update the molecular representation and learn the function that maps these graphs to properties obtained from reference electronic structure calculations such as Density Functional Theory (DFT).

There has been rapid progress in the development of GNN architectures for predicting material properties, such as such as SchNet, Atomistix, Crystal Graph Convolutional Neural Network (CGCNN), MatErials Graph Network (MEGNet), Atomistic Line Graph Neural Network (ALIGNN) and similar variants. These models consider only the pairwise interactions between bonded atoms or between atoms within a cut-off radius of typically 6 Å to 8 Å. Some have also incorporated many-body relationships, such as bond-angles, into the molecular representation. Nevertheless, all of these GNN models developed so far, can be categorised as local methods and are limited to analysing only the local environment around atoms or relying on multiple message passing layers to approximate long range interactions.

However, for certain systems and/or tasks, long range interactions can be important. One of the most eminent examples of long-range interactions is electrostatics, which, together with van der Waals, also define non-bonded interactions in the potential energy (PE) equation (Eq 1). To date there are only a few ML models that have incorporated electrostatic interactions into...
The electrostatic interaction between two atoms in a structure can be obtained using Coulomb’s law, shown in Eq. 2:

\[
E_{\text{potential}} = E_{\text{bonded}} + E_{\text{non-bonded}}
\]

\[
E_{\text{bonded}} = E_{\text{stretching}} + E_{\text{angle}} + E_{\text{dihedral}}
\]

\[
E_{\text{non-bonded}} = E_{\text{electrostatic}} + E_{\text{vanderWaals}}
\]

\[
E_{\text{electrostatic}} = \frac{1}{4\pi\varepsilon_0} \frac{q_i q_j}{r^2}
\]

where \(q_i\) and \(q_j\) are the atomic partial charges of atoms \(i\) and \(j\), \(\varepsilon_0\) is the permittivity of free space, and \(r\) is the distance between the two atoms. However, computing Coulomb interactions without truncation is limited by the availability of atomic partial charges. However, in 2012, Rupp et al. introduced the Coulomb Matrix, which includes a simplified representation of the Coulomb repulsion between two atoms, that doesn't make use of the atomic partial charges.

With the aim of enhancing GNN architectures akin to ALIGNN for the incorporation of long-range interactions, this paper introduces the Molecular Graph Representation (MGR) and the Molecular Graph Transformer (MGT). In this endeavour, the simplified Coulomb interactions that can be obtained from the Coulomb Matrix are explicitly included within the MGR and subsequently analysed by the MGT. The MGR splits the graphical representation of the system into three graphs: local graph \((G_{\text{local}})\), line graph \((G_{\text{line}})\), and fully connected graph \((G_{\text{global}})\).

The MGT alternates between graph attention layers on the \(G_{\text{global}}\) and graph convolutions on the \(G_{\text{line}}\) and \(G_{\text{local}}\), to update the molecular representation through non-bonding, many-body and two-body information. Our model is trained on both the MatBench\(^{31}\) and the QMOF\(^{32}\) to predict energetic, electronic and vibrational properties of solid-state materials directly from their unrelaxed structures.

2 Molecular Graph Transformer

2.1 Molecular Graph Representation

The \(G_{\text{local}}, G_{\text{line}}, \) and \(G_{\text{global}}\) sub-graphs of the MGR are used to represent both the bonded and non-bonded interactions between atoms. The local and line graph both describe bonded interactions, with the local graph describing pair-wise interactions (red edges in Figure 1), and the line graph describing 3-body interactions using angles between triplets of atoms (green edge in Figure 1). These two graphs can be considered the equivalent of \(E_{\text{stretching}}\) and \(E_{\text{angle}}\) in the PE equation. The full graph is based on the Coulomb Matrix representation, and it is used to represent the non-bonded interactions between pairs of atoms (blue edges in Figure 1), making it the equivalent of the \(E_{\text{electrostatic}}\) term in Equation 1.

The construction of the local and line graphs is the same as the ALIGNN\(^{13}\) representation. The local graph is constructed using a periodic 12-nearest-neighbour methodology, in which an edge is formed between an atom and its 12 nearest neighbours within...
a cut-off distance of 8 Å. Each atom is then assigned a nine features feature set based on its atomic species: 1) electronegativity; 2) covalent radius; 3) valence electrons; 4) group number; 5) electron affinity; 6) first ionization energy; 7) period number; 8) block number; 9) atomic volume. The feature sets are then encoded, through one-hot encoding, to form the feature vectors of the atoms. The edges are instead initialized with the distance between the two atoms that they connect. To form the feature vectors for the edges, a Radial Basis Function (RBF) is used with limits: 0 Å and 8 Å. The local graph can then be defined as 

\[ G_{\text{local}} = (h, e), \]

where \( h \) are nodes and \( e \) are edges between pair of atoms, and \( G_{\text{local}} \) has associated feature sets \( H = \{h_1, ..., h_l, ..., h_N\} \) and \( E = \{e_{ij}, e_{ik}, e_{ikl}, e_{mi}, \ldots \} \), where \( h_i \) is the feature vector given to node \( i \) and \( e_{ij} \) is the feature vector of the edge connecting nodes \( i \) and \( j \).

The line graph is derived from the local graph. Each node in the line graph represents an edge in the local graph, and nodes and corresponding edges share the same feature vector, such that any update on a node of the line graph is reflected on the corresponding edge in the local graph. Edges in the line graph, correspond to the relationship between pairs of edges in the local graph that have one atom in common, representing a three-body interaction system between triplets of atoms, i.e. bond pair \( e_{ij}, e_{ik} \) and atom triplet \( h_i, h_j, h_k \) where atom \( h_i \) is the shared atom. The line graph edge features are given by an RBF expansion of the angle formed by two connected local graph edges, shown in green in Figure 1. The line graph can then be defined as \( G_{\text{line}} = (e, t) \), where \( e \) are local graph edges and \( t \) are angles between connected edges or atom triplets, and \( G_{\text{line}} \) has associated feature sets \( E = \{e_{ij}, e_{ik}, e_{ikl}, e_{mi}, \ldots \} \) and \( T = \{t_{ij}, t_{ik}, t_{ikj}, t_{ijm}, \ldots \} \).

The full graph is constructed similarly to the local graph. Each node in the local graph represents an atom in the structure, and it shares its latent representation with the nodes of the local graph. Edges in the full graph represent an interaction between pairs of atoms, and they are formed between all atoms that are within a cut-off distance from each other. Full graph edges features are

![Diagram](https://doi.org/10.26434/chemrxiv-2024-rxwbc)

**Fig. 2** (a) MGT Encoder architecture. For the current layer \( L \) of the model, the node \( \{H^L\} \) embeddings are first updated using Multi-Headed Attention (MHA). The node features of central atoms \( h_i \) and neighbouring atoms \( h_j \) are separated and updated through individual linear layers. Attention scores are computed using atoms \( i \), \( j \) and coulomb repulsions \( \{F^L\} \) between them, and used to select features of atoms \( j \). Selected features of atoms \( j \) are added to core atoms \( i \) to update their features and further refined through residual and normalization steps. The updated node features \( H^L \) from the MHA block are then used as input to the ALIGNN block, which uses a series of Edge Gated Graph Convolutions (EGGC) to update the input node features \( H^L \) using edge information from the line graph \( T^L \) and local graph \( E^L \). The ALIGNN block, returns updated node features \( H^{L'} \), updated edge features \( E^{L'} \) and updated triplet features \( T^{L'} \). Node features \( H^{L'} \) are then further refined through \( N \) EGGC layers using edge features \( E^{L'} \). Lastly the output features from the EGGC layers, \( H^{L''} \), are then refined through a Linear Block, which passes them through two linear layers with a SiLU activation function between them. (b) Edge Gated Graph Convolution (EGGC) module. Each EGGC module splits its input node features \( \{X\} \) into core nodes \( x_c \) and neighbouring nodes \( x_n \) and are refined through separate linear layers. Using edge features \( e_{ij} \), core nodes \( i \) and neighbouring nodes \( j \), edge messages are computed and then added to core atoms \( i \) to update their representation. The EGGC then returns the edge message as an updated edge information \( \{E'\} \) and the updated node informations \( \{X'\} \).
derived from the Coulomb matrix of the structure and they represent the Coulomb repulsion between the two different atoms as described in the Coulomb matrix in Equation (3):

\[ f_{ij} = \begin{cases} 0.5Z_i^2Z_j^4 & \forall i = j \\ \frac{ZZ_i}{R_{ij}} & \forall i \neq j \end{cases} \]

where \( Z_i \) and \( Z_j \) are the atomic numbers of atoms \( i \) and \( j \) respectively, and \( R_{ij} \) is the distance between the two atoms. The full graph can then be defined as \( G_{\text{global}} = (h, f) \), where \( h \) are nodes and \( f \) are edges between pairs of atoms. \( G_{\text{global}} \) has associated feature sets \( H = \{ h_1, ..., h_i, ..., h_N \} \) and \( F = \{ f_{ij}, f_{ik}, ... \} \), where \( h_i \) is the feature vector given to node \( i \) and \( f_{ij} \) is the feature vector of the edge connecting nodes \( i \) and \( j \).

### 2.2 Molecular Encoder

The main part of the MGT is the encoder module. This module executes updates on the nodes and edges of the MGR, by applying different update functions based on the sets of edges provided by the three parts of the MGR. Using the edges provided by the \( G_{\text{global}} \), the module updates the nodes using Multi-Headed Attention (MHA). The encoder, then, uses a series of Edge Gated Graph Convolution (EGGC) modules, which are aggregated into a block called the ALIGNN Block, to update the edges and nodes of both the \( G_{\text{line}} \) and the \( G_{\text{local}} \). Outside of the ALIGNN Block, the EGGC module is also used for a further update to the edges and nodes of the \( G_{\text{local}} \). Lastly, the encoder also performs a final update on the nodes without using any edge information by running the nodes through a linear block, which, contains a fully connected layer with a SiLU activation function, followed by another fully connected layer.

The modifications to the feature vectors follow a global-to-local sequence. Initially, nodes for increased attention are determined through non-bonding two-body interactions from the global graph, followed by updates using the same interactions. Subsequently, utilizing the line graph, updates involving three-body interactions are executed, succeeded by updates involving two-body interactions from the local graph. Lastly, updates based on single-body information are performed using only the node information.

### Multi-Headed Attention

The MHA block, in Figure 2a, is derived from the Transformer attention mechanism and it is adapted to a message passing architecture. Given a input set of node features \( H^L = h_1^L, h_2^L, ..., h_N^L \) at layer \( L \), it first splits the set into \( Q, K, V \), using learnable parameters \( W_Q, W_K, W_V \). \( Q \) corresponds to the target nodes \( h_i \) (nodes to be updated) and \( K \) and \( V \) correspond to the neighbouring nodes \( h_j \). \( Q \), \( K \), and \( V \) are then split into \( M \) different subsets (heads), \( Q = \{ Q^1, Q^2, ..., Q^M \}, K = \{ K^1, K^2, ..., K^M \}, \)

\[ V = \{ V^1, V^2, ..., V^M \} \], such that we can obtain attention scores for each subset (head). If the edges of the input graph contain edge attributes, then, given a set of edge features \( F \) at layer \( L \), they are transformed into a set \( F \) using learnable parameter \( W_F \) and then split into \( M \) different subsets (heads) as well, \( F = \{ F^1, F^2, ..., F^M \} \).

In the case of one attention head, we have \( Q^m = Q, K^m = K, V^m = V \) and \( F^m = F \). For each node \( h_i \), the attention for each subset of \( Q, K, V \) and \( F \) is obtained as:

\[ Q = W_Q h_i, \quad K = W_K h_j, \quad V = W_V h_j, \quad F = W_F f_{ij} \quad (4) \]

\[ S_{ij}^m = \frac{Q^m K^m + F^m}{\sqrt{d_m}} \quad (5) \]

\[ \text{Att}^m(h_i) = \sum_{j \in N_i} \text{softmax}(S_{ij}^m)V^m \quad (6) \]

where \( d_m \) is the dimension of each subset of \( Q, K, V, F \), \( S_{ij}^m \) is the attention score between nodes \( h_i \) and \( h_j \), \( N_i \) defines the set of neighbours of \( h_i \), \( \sum_{j \in N_i} \) denotes the summation with all the neighbours of node \( h_i \), and \( \text{softmax} \) is used to normalize each message along the edges of node \( h_i \) such that:

\[ \text{softmax}(S_{ij}^m) = \frac{\exp(S_{ij}^m)}{\sum_{k \in N_i} \exp(S_{ik}^m)} \quad (7) \]

The update to each node is then defined as:

\[ \hat{h}_i = W_H h_i + \frac{M}{\text{m=1}} \text{Att}^m(h_i) \quad (8) \]

\[ h_i' = \text{Norm}(\hat{h}_i) \quad (9) \]

where \( \text{Norm} \) denotes concatenation, and \( W_H \) is a learnable parameter.

### Edge Gated Graph Convolution

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder Layers</td>
<td>2</td>
</tr>
<tr>
<td>MHA Layers</td>
<td>1</td>
</tr>
<tr>
<td>ALIGNN Blocks</td>
<td>3</td>
</tr>
<tr>
<td>EGGC Layers</td>
<td>1</td>
</tr>
<tr>
<td>Atom Input Features</td>
<td>90</td>
</tr>
<tr>
<td>Edge Input Features</td>
<td>80</td>
</tr>
<tr>
<td>Angle Input Features</td>
<td>40</td>
</tr>
<tr>
<td>Coulomb Input Features</td>
<td>120</td>
</tr>
<tr>
<td>Embedding Features</td>
<td>256</td>
</tr>
<tr>
<td>Hidden Features</td>
<td>512</td>
</tr>
<tr>
<td>FC Layer Features</td>
<td>128</td>
</tr>
<tr>
<td>Global Pooling Function</td>
<td>Average</td>
</tr>
<tr>
<td>Batch Size per GPU</td>
<td>2</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Table 1: Optimal MGT configuration and hyper-parameters obtained through testing. This configuration and hyper-parameters were used for the testing of the model on both datasets used in this paper.
The EGGC used in this work, shown in Figure 2b, was introduced by Choudhary and DeCost\textsuperscript{13}. It takes inspiration from the CGCNN update\textsuperscript{11} but in contrast to it, the edge features are incorporated into normalized edge gates. Furthermore, EGGC, unlike the CGCNN update, also updates edge features by utilising edge messages. Using EGGC, the input node representations $H'$ are updated as follows:

$$h_i' = h_i' + \text{SiLU}(\text{Norm}(W_{src}h_i' + \sum_{j \in N_i} \hat{e}_{ij}' W_{dst}h_j'))$$

$$\hat{e}_{ij} = \frac{\sigma(e_{ij})}{\sum_{k \in N_j} \sigma(e_{ik}) + \epsilon}$$

$$e_{ij}' = e_{ij} + \text{SiLU}(\text{Norm}(Ah_i' + Bh_j' + Ce_{ij})))$$

where SiLU is the Sigmoid-weighted Linear Unit\textsuperscript{14}, $\sigma$ denotes the sigmoid function and $A, B, C$ are weight matrices for updating $h_i', h_j'$ and $e_{ij}$ respectively.

**ALIGNN Block**

The ALIGNN Block combines an EGGC update on the line graph $G_{line}$ with an EGGC update on the local graph $G_{local}$. The convolution on $G_{line}$ produces updates edge updates that are propagated to $G_{local}$, which, further updates the edge features and the atom features.

$$m', t' = EGGC(G_{line}, e, t)$$

$$h', e' = EGGC(G_{local}, h, m')$$

### 2.3 Overall Model Architecture

The MGT, shown in Figure 3, is composed of $M$ encoder layers, with a pooling function, applied on the edge features of all three graphs, in between the encoders. After $M$ encoder layers we then apply a global pooling function to aggregate the node features into one feature for the whole graph. Finally to predict the properties of the input structure or perform classification on it, we apply a fully connected regression or classification layer. Table 1 shows the hyper-parameters that were used to train the model with which we obtained the results shown in the Results and Discussion section. These hyper-parameters were obtained through hypothesis-driven hyper-parameter search.

### 2.4 Model Implementation and Training

The MGT is implemented using PyTorch\textsuperscript{35} and the Deep Graph Library\textsuperscript{15}. The code also relies on Pytorch Fabric for the distribution of the model across multiple GPUs and devices. For regression tasks the loss is obtained using the Mean Squared Error (MSE) function, while the error is obtained using the Mean Average Error (MAE) function. The models trained on the QMOF database were trained for 100 epochs, while 300 epochs were used for those trained on the MatBench database. For all models the Adam optimizer was used with a weight decay of $10^{-5}$ and a learning rate of 0.0001. Calculations were performed using the Sulis Tier 2 HPC platform and on the JADE 2 Tier 2 HPC platform. On the Sulis HPC at most 5 nodes with 3 A100 40GB GPUs were used, while on JADE 2 the model was trained on 8 V100 32GB GPUs.

### 3 Results and Discussion

#### 3.1 Performance on Datasets

The MGT was created with the purpose of predicting properties of solid-state materials from their unrelaxed structures. To evaluate its performance, the Materials Project's MatBench\textsuperscript{31} version 0.1 dataset and the QMOF\textsuperscript{32} version 14 database were used.

The MatBench\textsuperscript{31} dataset encompasses eight distinct regression tasks, each associated with an individual database containing a varied number of structures, ranging from 636 for the "jdf2d" task to 106,113 for the "Formation Energy" task. The "jdf2d" task involves predicting the exfoliation energy for separating 2D lay-
Table 2 Performance on the MatBench v0.1 dataset. The errors shown in the table are Mean Average Errors (MAE). For the models other than the MGT the values have been obtained from the official MatBench website. Next to the errors, in the parenthesis, their respective rankings for each task are shown.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>jdt2d</th>
<th>phonons</th>
<th>dielectric</th>
<th>log10 GVRH</th>
<th>log10 KVRH</th>
<th>perovskites</th>
<th>bandgap</th>
<th>formation E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Units</td>
<td>meV/atom</td>
<td>cm⁻¹</td>
<td>unitless</td>
<td>log₁₀ GPa</td>
<td>log₁₀ GPa</td>
<td>eV/unit cell</td>
<td>eV</td>
<td>eV/atom</td>
</tr>
<tr>
<td>ALIGNN</td>
<td>43.4244 (8)</td>
<td>29.5385 (3)</td>
<td>0.3449 (13)</td>
<td>0.0715 (3)</td>
<td>0.0568 (5)</td>
<td>0.0288 (2)</td>
<td>0.1861 (3)</td>
<td>0.0215 (3)</td>
</tr>
<tr>
<td>AMMExpress v2020</td>
<td>39.8497 (6)</td>
<td>56.1706 (13)</td>
<td>0.3150 (6)</td>
<td>0.0874 (10)</td>
<td>0.0647 (9)</td>
<td>0.2005 (12)</td>
<td>0.2824 (13)</td>
<td>0.1726 (15)</td>
</tr>
<tr>
<td>CGCNN v2019</td>
<td>49.2440 (13)</td>
<td>57.7635 (14)</td>
<td>0.5988 (15)</td>
<td>0.0895 (11)</td>
<td>0.0712 (12)</td>
<td>0.0452 (9)</td>
<td>0.2972 (14)</td>
<td>0.0337 (7)</td>
</tr>
<tr>
<td>coNGN</td>
<td>36.1698 (5)</td>
<td>28.8874 (2)</td>
<td>0.3142 (5)</td>
<td>0.0670 (1)</td>
<td>0.0491 (1)</td>
<td>0.0290 (3)</td>
<td>0.1697 (2)</td>
<td>0.0178 (2)</td>
</tr>
<tr>
<td>coGN</td>
<td>37.1652 (4)</td>
<td>29.7117 (4)</td>
<td>0.3088 (4)</td>
<td>0.0689 (2)</td>
<td>0.0535 (2)</td>
<td>0.0269 (1)</td>
<td>0.1559 (1)</td>
<td>0.0170 (1)</td>
</tr>
<tr>
<td>CrabNet</td>
<td>45.6104 (9)</td>
<td>55.1114 (12)</td>
<td>0.3234 (9)</td>
<td>0.1014 (14)</td>
<td>0.0758 (13)</td>
<td>0.4065 (14)</td>
<td>0.2655 (12)</td>
<td>0.0862 (13)</td>
</tr>
<tr>
<td>DimeNet++ (kgcnn v2.1.0)</td>
<td>49.0243 (12)</td>
<td>37.4619 (6)</td>
<td>0.3400 (12)</td>
<td>0.0792 (6)</td>
<td>0.0572 (6)</td>
<td>0.0376 (8)</td>
<td>0.1993 (5)</td>
<td>0.0235 (5)</td>
</tr>
<tr>
<td>Finder_v1.2 composition only version</td>
<td>47.9614 (11)</td>
<td>46.5751 (10)</td>
<td>0.3204 (8)</td>
<td>0.0996 (13)</td>
<td>0.0764 (14)</td>
<td>0.6450 (16)</td>
<td>0.2308 (10)</td>
<td>0.0839 (12)</td>
</tr>
<tr>
<td>Finder_v1.2 structure based version</td>
<td>46.1339 (10)</td>
<td>50.7406 (11)</td>
<td>0.3197 (7)</td>
<td>0.0910 (12)</td>
<td>0.0693 (11)</td>
<td>0.0320 (4)</td>
<td>0.2193 (7)</td>
<td>0.0343 (8)</td>
</tr>
<tr>
<td>MegNet (kgcnn v2.1.0)</td>
<td>54.1719 (15)</td>
<td>28.7606 (1)</td>
<td>0.3391 (11)</td>
<td>0.0871 (9)</td>
<td>0.0668 (10)</td>
<td>0.0352 (6)</td>
<td>0.1934 (4)</td>
<td>0.0252 (6)</td>
</tr>
<tr>
<td>MODNet (v0.1.12)</td>
<td>33.1918 (2)</td>
<td>34.2751 (5)</td>
<td>0.2711 (1)</td>
<td>0.0731 (4)</td>
<td>0.0548 (3)</td>
<td>0.0908 (10)</td>
<td>0.2199 (8)</td>
<td>0.0448 (10)</td>
</tr>
<tr>
<td>MODNet (v0.1.10)</td>
<td>34.5368 (3)</td>
<td>38.7524 (7)</td>
<td>0.2970 (2)</td>
<td>0.0731 (5)</td>
<td>0.0548 (4)</td>
<td>0.0908 (11)</td>
<td>0.2199 (9)</td>
<td>0.0448 (11)</td>
</tr>
<tr>
<td>RF-SCM/Magpie</td>
<td>50.0440 (14)</td>
<td>67.6126 (15)</td>
<td>0.4196 (14)</td>
<td>0.1040 (15)</td>
<td>0.0820 (15)</td>
<td>0.2355 (13)</td>
<td>0.3452 (15)</td>
<td>0.1165 (14)</td>
</tr>
<tr>
<td>SchNet (kgcnn v2.1.0)</td>
<td>42.6637 (7)</td>
<td>38.9636 (8)</td>
<td>0.3277 (10)</td>
<td>0.0796 (7)</td>
<td>0.0590 (7)</td>
<td>0.0342 (5)</td>
<td>0.2352 (11)</td>
<td>0.0218 (4)</td>
</tr>
<tr>
<td>MGT</td>
<td>31.4223 (1)</td>
<td>39.0179 (9)</td>
<td>0.3047 (3)</td>
<td>0.0840 (8)</td>
<td>0.0636 (8)</td>
<td>0.0361 (7)</td>
<td>0.2145 (6)</td>
<td>0.0378 (9)</td>
</tr>
<tr>
<td>Dummy</td>
<td>67.2851 (16)</td>
<td>323.9822 (16)</td>
<td>0.8088 (16)</td>
<td>0.2931 (16)</td>
<td>0.2897 (16)</td>
<td>0.5660 (15)</td>
<td>1.3272 (16)</td>
<td>1.0059 (16)</td>
</tr>
</tbody>
</table>
ers from crystal structures, computed with the OptB88vdW and TBmB exchange-correlation DFT functionals. The "Phonons" task is dedicated to predicting the vibrational properties of crystal structures computed using the ABINIT with the harmonic approximation based on density functional perturbation theory. The "Dielectric" task is concerned with predicting refractive index from crystal structures. The "log10 GVRH" and "log10 KVRH" tasks involve predicting the logarithm (base 10) of the shear (GVRH) and bulk (KVRH) modulus property of crystal structures. The "Bandgap" task focuses on the prediction of the electronic bandgap of crystal structures computed with the Perdew Burke Ernzerhof (PBE) functional. Lastly the "Perovskites" and "Formation Energy" tasks are dedicated to predicting the formation energy of crystal structures, with the "Perovskites" task focusing on perovskites. For all the tasks, the train-test splits provided by the MatBench api were used, and the train set was further divided into train-validation splits of approximately 80%-20% for each task. The performance of the MGT model on the MatBench dataset, is shown in Table 2.

The "jdft2d" and "Dielectric" tasks have benefited the most from the inclusion of electrostatic interactions with the MGT model, with improvements of 27% and 12%, respectively, compared to ALIGNN. The jdft2d task is related to the prediction of exfoliation energy of crystal structures, which involves the energy required to remove a layer of the material from its surface. Since molecular layers usually are connected through non-covalent weak interactions as in the case of graphene, the inclusion of non-bonding interactions, such as electrostatics, can benefit GNN models in this task. Inclusion of electrostatics can also benefit in the dielectric task, which predicts the refractive index and is affected by the electrostatic interactions between all neighbouring atoms within 12 in the structure.

On the other hand, the "Formation Energy" task has seen the least benefit from this inclusion, with a MAE of 0.0448 eV/atom with the MGT model compared to 0.0215 meV/atom with ALIGNN. The jdft2d task is related to the prediction of exfoliation energy of crystal structures, which involves the energy required to remove a layer of the material from its surface. Since molecular layers usually are connected through non-covalent weak interactions as in the case of graphene, the inclusion of non-bonding interactions, such as electrostatics, can benefit GNN models in this task. Inclusion of electrostatics can also benefit in the dielectric task, which predicts the refractive index and is affected by the electrostatic interactions between all neighbouring atoms within 12 in the structure.

The "jdft2d" and "Dielectric" tasks have benefited the most from the inclusion of electrostatic interactions with the MGT model, with improvements of 27% and 12%, respectively, compared to ALIGNN. The jdft2d task is related to the prediction of exfoliation energy of crystal structures, which involves the energy required to remove a layer of the material from its surface. Since molecular layers usually are connected through non-covalent weak interactions as in the case of graphene, the inclusion of non-bonding interactions, such as electrostatics, can benefit GNN models in this task. Inclusion of electrostatics can also benefit in the dielectric task, which predicts the refractive index and is affected by the electrostatic interactions between all neighbouring atoms within 12 in the structure.

Overall, the results on both datasets suggest that the attention module can be a valuable tool for analysing long-range interactions and improving the performance of graph neural networks.

3.2 Ablation Study

Each component of the MGT Encoder is ablated to further understand the impact that they have on the performance of the model. The ablation study was performed using the QMOF dataset for the prediction of Bandgap, HOMO and LUMO energies. All the parameters, other than the number of MHA, ALIGNN and EGGC Layers, were kept the same as the ones specified in Table 1.

The QMOF dataset consists of 20375 Metal Organic Framework (MOF) structures with results from electronic structure calculations done using the functionals: PBE, High Local Exchange 2017 (HLE17) and Heyd-Scuseria-Ernzerhof 06 (HSE06) with 10% and 25% of the Hartree-Fock exact exchange. A train-validation-test split of 16000-2000-2375 was applied and the performance of the MGT on this dataset is shown in Table 3. The performance on the prediction of bandgap values in the QMOF database is similar to that observed in the bandgap task in the MatBench dataset, however, we also have two additional properties, the Highest Occupied Molecular Orbital (HOMO) and the Lowest Unoccupied Molecular Orbital (LUMO) energy levels. On all three properties, the MGT demonstrates a difference in error of about 0.02 eV with respect to ALIGNN.

Table 3 Performance on the QMOF v14 dataset. All the error reported have been obtained by retraining the original models on the QMOF v14 dataset, using the train-validation-test splits reported in this paper.

<table>
<thead>
<tr>
<th>Property</th>
<th>MGT</th>
<th>ALIGNN</th>
<th>CGCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandgap (eV)</td>
<td>0.240</td>
<td>0.224</td>
<td>0.330</td>
</tr>
<tr>
<td>HOMO (eV)</td>
<td>0.263</td>
<td>0.245</td>
<td>0.361</td>
</tr>
<tr>
<td>LUMO (eV)</td>
<td>0.252</td>
<td>0.232</td>
<td>0.330</td>
</tr>
</tbody>
</table>

3.2 Ablation Study

Each component of the MGT Encoder is ablated to further understand the impact that they have on the performance of the model. The ablation study was performed using the QMOF dataset for the prediction of Bandgap, HOMO and LUMO energies. All the parameters, other than the number of MHA, ALIGNN and EGGC Layers, were kept the same as the ones specified in Table 1.

Table 4 Performance of each component (MHA, ALIGNN, EGGC) of the MGT Encoder on their own, at varying number of repetitions. In the tests performed for each component, the linear block is included after the component being tested, while the other two components were excluded completely.

<table>
<thead>
<tr>
<th>Component</th>
<th>Number of Repetitions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>MHA</td>
<td>0.4031</td>
</tr>
<tr>
<td>ALIGNN</td>
<td>0.3224</td>
</tr>
<tr>
<td>EGGC</td>
<td>0.3840</td>
</tr>
</tbody>
</table>

Excluding all three modules (MHA, ALIGNN and EGGC) the model has an error of 0.8734 eV, including even just a single MHA shows an improvement of at least 54% bringing the error down to 0.4031, which demonstrates the importance of these layers.

**Table 4 Performance of each component (MHA, ALIGNN, EGGC) of the MGT Encoder on their own, at varying number of repetitions. In the tests performed for each component, the linear block is included after the component being tested, while the other two components were excluded completely.**
Excluding two and using just one of the modules shows the individual performance of these layers. Using only EGGC layers there is an improvement of at least 56% over using no layers, and the performance saturates at 4 layers with an error of 0.3016 eV. Performance using only one ALIGNN layer is improved to 0.3224 eV and saturates at 4 layers with an error of 0.2609 eV. Meanwhile, the use of MHA Layers only shows a performance saturation at 4 layers with an error of 0.3816 eV.

The effect of each layer and the coupling between them can also be studied by varying the number of layers, while using all modules at the same time. Due to the number of possible configurations and the training time only a subset of them have been tested.

Increasing the number of MHAs within an encoder has almost no effect on the performance on the model; using configurations with 1, 2, 3 and 4 MHAs the MAE obtained are 0.2888 eV, 0.2880 eV, 0.2865 eV, 0.2877 eV respectively, which shows a very small improvement when using more MHAs with a performance saturation at 3 Layers. Increasing the number of ALIGNN Blocks, on the other hand has the biggest effect on the QMOF, with errors of 0.2888 eV, 0.2685 eV, 0.2661 eV, 0.2672 eV using 1, 2, 3, 4 Layers respectively, showing improvements with an increase of Layers up to 3. Increasing the number of EGGCs, similarly to MHA, also brings small improvements on the performance, with errors of 0.2888 eV, 0.2828 eV, 0.2750 eV and 0.2716 eV using 1, 2, 3, and 4 Layers respectively. These results have been obtained by changing the number of repetitions of each component while keeping the other two components at one. Nevertheless, even when testing for all possible combinations the results (shown in Table S2 in the supporting information) are almost the same.

Changing the number of layers of each module, impacts not only the performance but also size, training time and inference time of the model. Although, getting access to more powerful computers is becoming easier, not everyone has the latest and best computing resources, thus, the decision to add more or less layers is also dependent on their impact upon computational requirements. From Figure 4a it can be seen that the ALIGNN blocks are the ones that have the biggest impact on the model size, with each block adding 2,630,656 parameters, while, the MHA and EGGC modules add 1,052,672 and 1,315,328 respectively. Nevertheless, the module that has the largest impact on training and inference times is the MHA module, as shown in Figure 4b and 4c. Each additional MHA layer adds around 20 seconds to the inference time, double that of each additional ALIGNN block and quadruple the EGGC Layers, which, add around 10 seconds and 5 seconds respectively.

4 Conclusions

In this paper the Molecular Graph Representation (MGR) and the Molecular Graph Transformer (MGT) were introduced and tested on the prediction of several materials properties. The combination of MGT and MGR introduces a methodology for including long-range electrostatic interactions between pairs of atoms within an arbitrary cut-off distance, here set at 12 Å, by using a simplified representation of Coulomb interactions obtained from the Coulomb Matrix\textsuperscript{30}. The MGT has achieved results in line with the state-of-the-art models on most tasks, and in some cases performing better than previously published models. While the size and training time of the model can be a constraint for some users, the MGT has shown capable of achieving great performance even on tasks with smaller datasets, such as the jdft2d task in the MatBench dataset, which contains only 636 structures. Therefore,
users have the option of reducing the training time by training on a smaller set. Furthermore, with the modularity of the model, its size can be reduced, making it trainable even on smaller machines, at the cost of reduced performance.

**Author Contributions**

All authors designed the model and the computational framework. M.A. carried out the implementation and performed the calculations. M.A. wrote the manuscript with input from all authors. G.S., D.D.T, and R.C.O. conceived the study and were in charge of overall direction and planning.

**Conflicts of interest**

There are no conflicts to declare.

**Acknowledgements**

M.A. acknowledges the mini-Centre-for-Doctoral-Training in CO2-Conversion at QMUL for PhD scholarship. Calculations were performed using the Sulis Tier 2 HPC platform hosted by the Scientific Computing Research Technology Platform at the University of Warwick and on the JADE 2 Tier 2 HPC platform hosted by the Hartree Center at the University of Oxford. Sulis is funded by EPSRC Grant EP/T022108/1 and the HPC Midlands+ consortium. JADE 2 is funded by EPSRC Grant.

**Notes and References**


