Journal Name

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

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

Marco Anselmi,[a] Greg Slabaugh,[b*], Rachel Crespo-Otero[c*] and Devis Di Tommaso[a*]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

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,[1] link and community detection on social media networks,[2–4] representation learning on textual graphs,[5,6] medical image analysis for disease detection[7–9] and property prediction for molecular and crystalline materials.[10–18]

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.[19] GNN models represent molecules or crystalline materials as graphs with a node for each constituent atom and an edges 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,[10] Crystal Graph Convolutional Neural Network (CGCNN),[11] MatErials Graph Network (MEGNet),[12] Atomistic Line Graph Neural Network (ALIGNN)[13] and similar variants.[14–18,20–24] 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.[13,14,16,17,20] Nevertheless, all of these GNN models developed so far, can be categorised as local methods[25] 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

[a] School of Physical and Chemical Sciences, Queen Mary University of London, Mile End Rd, Bethnal Green, London, E1 4NS, United Kingdom.
[b] Digital Environment Research Institute, Queen Mary University of London, Empire House, 67-75 New Road, London, E1 1HH, United Kingdom.
[c] Department of Chemistry, University of College London, 20 Gordon Street, London WC1H 0AJ

their architecture.[26–29] The electrostatic interaction between two atoms in a structure can be obtained using Coulomb's law, shown in Eq 2.

$$\begin{aligned} E_{potential} &= E_{bonded} + E_{non-bonded} \\ E_{bonded} &= E_{stretching} + E_{angle} + E_{dihedral} \\ E_{non-bonded} &= E_{electrostatic} + E_{vanderWaals} \end{aligned} \quad (1)$$

$$E_{electrostatic} = \frac{1}{4\pi\varepsilon_0}\frac{q_i q_j}{r^2} = k_e\frac{q_i q_j}{r^2} \quad (2)$$

where q_i and q_j are the atomic partial charges of atoms i and j, ε_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,[30] 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[13] 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[30] 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_{local}), line graph (G_{line}), and fully connected graph (G_{global}). The MGT alternates between graph attention layers on the G_{global} and graph convolutions on the G_{line} and G_{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_{local}, G_{line}, and G_{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 $E_{stretching}$ and E_{angle} in the PE equation. The full graph is based on the Coulomb Matrix[30] 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 $E_{electrostatic}$ 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

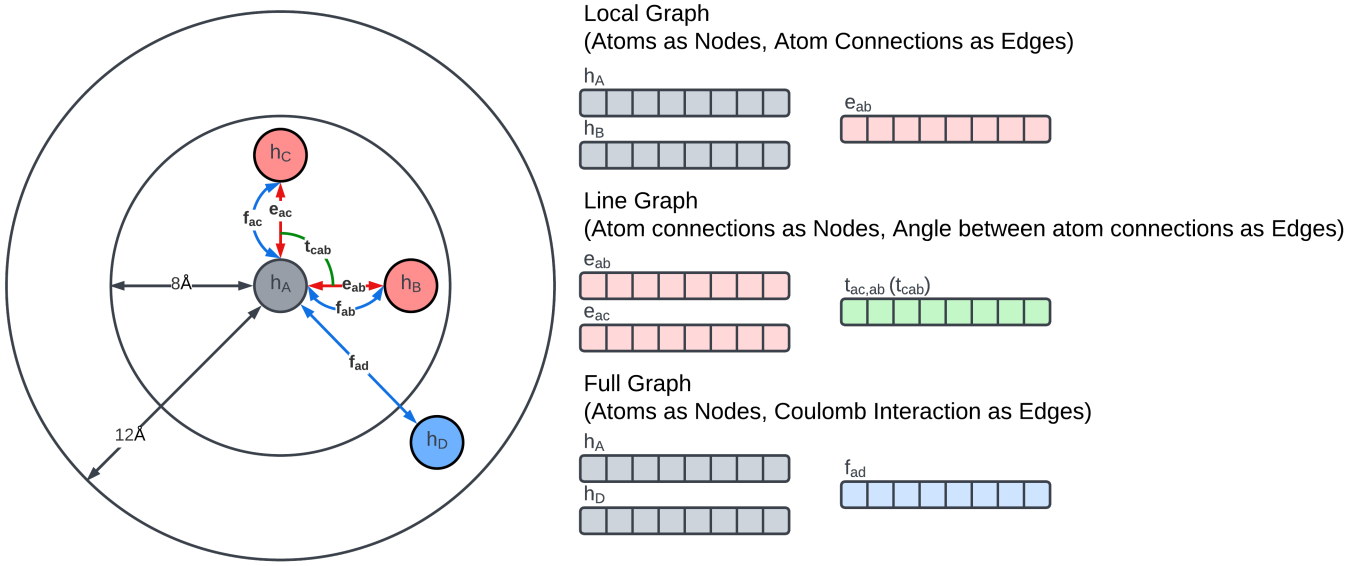
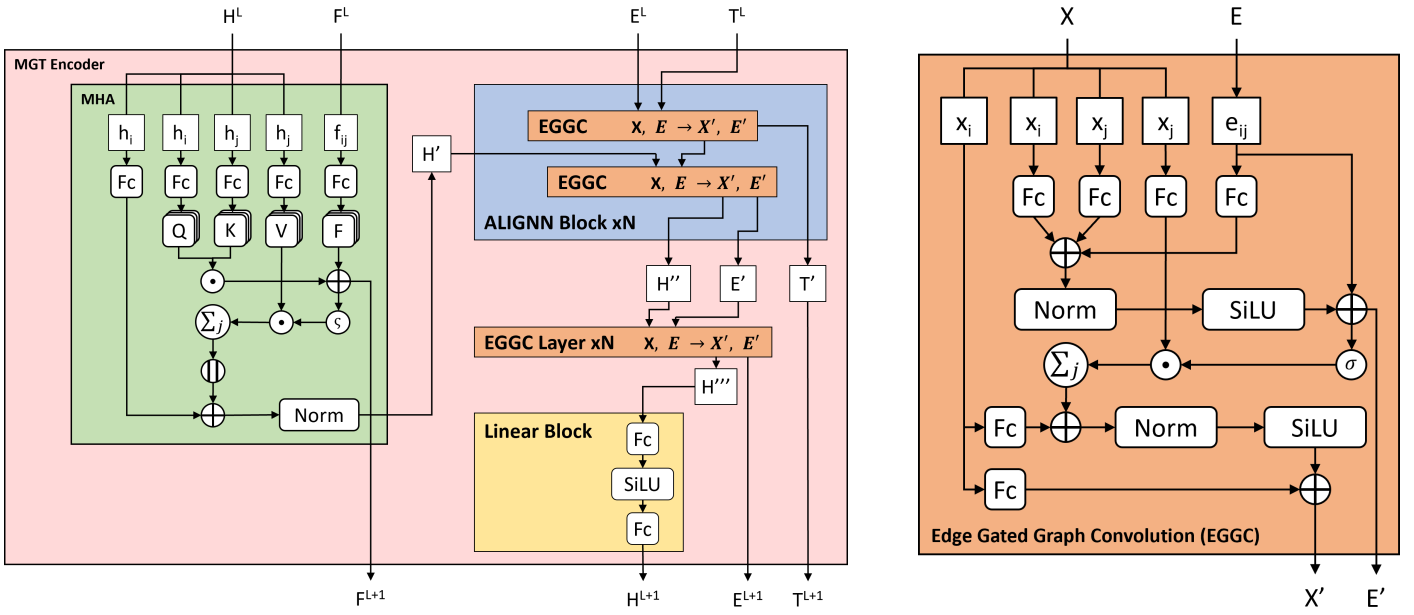


Fig. 1 Schematic showing the Molecular Graph Representation. For simplicity, only an atom connected to three other atoms is shown. Molecules are encoded by converting each atom i into a vector representation h_i. Using the Local Graph interactions between atoms i, j are dependent on their representations h_i, h_j and the distance between them e_{ij}. While, using the Line Graph interactions between triplets of atoms i, j, k are dependent on the distance between pairs of atoms e_{ij}, e_{jk} and the angle between the atoms $t_{ij,jk}(t_{ijk})$. Lastly, using the Full Graph the interaction between pairs of atoms I, J are dependent on their vector representation h_i, h_j and the Coulomb repulsion between them f_{ij}.

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 $G_{local} = (h, e)$, where h are nodes and e are edges between pair of atoms, and G_{local} has associated feature sets $H = \{h_1, ..., h_i, ..., h_N\}$ and $E = \{e_{ij}, e_{ik}, e_{kl}, e_{mi}, ...\}$, 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 $G_{line} = (e, t)$, where e are local graph edges and t are angles between connected edges or atom triplets, and G_{line} has associated feature sets $E = \{e_{ij}, e_{ik}, e_{kl}, e_{mi}, ...\}$ and $T = \{t_{ijk}, t_{ikl}, t_{ijm}, ...\}$.

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



(a) (b)

Fig. 2 (a) MGT Encoder layer 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' 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' using edge information from the line graph T^L and local graph E^L. The ALIGNN Block, returns updated node features H'', updated edge features E' and updated triplet features T'. Node features H'' are then further refined through N EGGC layers using edge features E'. Lastly the output features from the EGGC layers, H''', are then refined through a Linear Block, which passes them through two linear layers with a SiLU activation function between then. (b) Edge Gated Graph Convolution (EGGC) module. Each EGGC module splits its input node features (X) into core nodes x_i and neighbouring nodes x_j 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[30] of the structure and they represent the Coulomb repulsion between the two different atoms as described in the Coulomb matrix in Equation 3

$$f_{ij} = \begin{cases} 0.5Z_i^{2.4} & \forall\, i = j \\ \frac{Z_i Z_j}{R_{ij}} & \forall\, i \neq j \end{cases} \tag{3}$$

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 $G_{global} = (h, f)$, where h are nodes and f are edges between pairs of atoms. G_{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.

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

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.

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_{global}, the module updates the nodes using Multi-Headed Attention (MHA). The encoder, then, uses a series of Edge Gated Graph Convolution (EGCC) modules, which are aggregated into a block called the ALIGNN Block, to update the edges and nodes of both the G_{line} and the G_{local}. Outside of the ALIGNN Block, the EGCC module is also used for a further update to the edges and nodes of the G_{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[33] 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^L 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:

$$Q = W_Q h_i, \quad K = W_K h_j, \quad V = W_V h_j, \quad F = W_F f_{ij} \tag{4}$$

$$S_{ij}^m = \frac{Q^m K^m + F^m}{\sqrt{d_m}} \tag{5}$$

$$Att^m(h_i) = \sum_{j \in N_i} softmax(S_{ij}^m) V^m \tag{6}$$

where d_m is the dimension of each subset of Q, K, V and 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, $\sum_{j \in H^L}$ denotes the summation with all the neighbours of node h_i, and $softmax$ is used to normalize each message along the edges of node h_i such that:

$$softmax(S_{ij}^m) = \frac{exp(S_{ij}^m)}{\sum_{k \in N_i} exp(S_{ik}^m)} \tag{7}$$

The update to each node is then defined as:

$$\hat{h}_i = W_H h_i + \Big\Vert_{m=1}^{M} Att^m(h_i) \tag{8}$$

$$h_i' = Norm(\hat{h}_i) \tag{9}$$

where $\|$ denotes concatenation, and W_H is a learnable parameter.

Edge Gated Graph Convolution

The EGGC used in this work, shown in Figure 2b, was introduced by Choudhary and DeCost [13]. It takes inspiration from the CGCNN update [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:

$$h_i'' = h_i' + SiLU(Norm(W_{src}h_i' + \sum_{j \in N_i} \hat{e}_{ij}' W_{dst} h_j')) \quad (10)$$

$$\hat{e}_{ij}' = \frac{\sigma(e_{ij}')}{\sum_{k \in N_i} \sigma(e_{ik}') + \in} \quad (11)$$

$$e_{ij}' = e_{ij} + SiLU(Norm(Ah_i' + Bh_j' + Ce_{ij})) \quad (12)$$

where $SiLU$ is the Sigmoid-weighted Linear Unit [34], σ 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.

$$m', t' = EGGC(G_{line}, e, t) \quad (13)$$

$$h', e' = EGGC(G_{local}, h, m') \quad (14)$$

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 [35] and the Deep Graph Library [36]. 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.

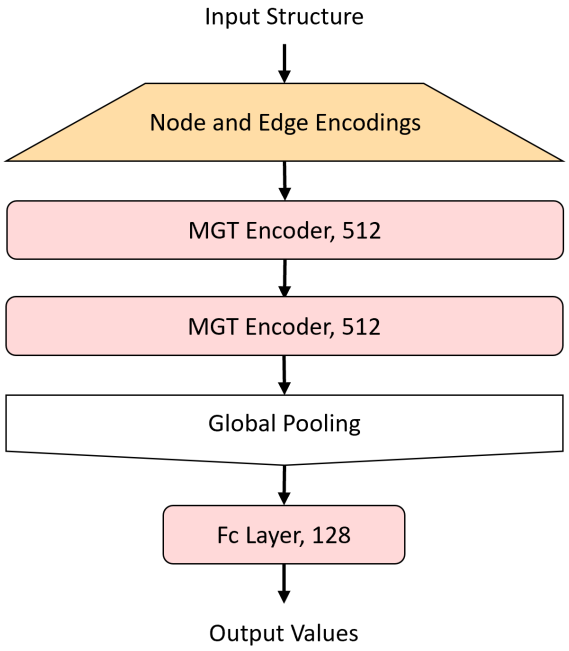


Fig. 3 Flowchart for the overall model architecture. The nodes and edges from the MGR are first transformed to feature vectors through encoding layers for the nodes and each edge type. The feature vectors are then passed through a series of MGT Encoder layers each with the same dimension. The output node features from the MGT Encoders are then aggregated using a pooling function to create a global feature vector for the whole structure. The global features are then refined through a linear layer, which, provides the output to the model. For this paper, 2 MGT Encoder were used, with an input and output feature length of 512.

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 [31] version 0.1 dataset and the QMOF [32] version 14 database were used.

The MatBench [31] dataset encompasses eight distinct regression tasks, each associated with an individual database containing a varied number of structures, ranging from 636 for the "jdft2d" task to 106,113 for the "Formation Energy" task. The "jdft2d" 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

Tasks	jdft2d	phonons	dielectric	log10 GVRH	log10 KVRH	perovskites	bandgap	formation E
Units	$meV/atom$	cm^{-1}	$unitless$	$\log_{10} GPa$	$\log_{10} GPa$	$eV/unit\ cell$	eV	$eV/atom$
ALIGNN	43.4244 (8)	29.5385 (3)	0.3449 (13)	0.0715 (3)	0.0568 (5)	0.0288 (2)	0.1861 (3)	0.0215 (3)
AMMExpress v2020	39.8497 (6)	56.1706 (13)	0.3150 (6)	0.0874 (10)	0.0647 (9)	0.2005 (12)	0.2824 (13)	0.1726 (15)
CGCNN v2019	49.2440 (13)	57.7635 (14)	0.5988 (15)	0.0895 (11)	0.0712 (12)	0.0452 (9)	0.2972 (14)	0.0337 (7)
coNGN	36.1698 (5)	28.8874 (2)	0.3142 (5)	0.0670 (1)	0.0491 (1)	0.0290 (3)	0.1697 (2)	0.0178 (2)
coGN	37.1652 (4)	29.7117 (4)	0.3088 (4)	0.0689 (2)	0.0535 (2)	0.0269 (1)	0.1559 (1)	0.0170 (1)
CrabNet	45.6104 (9)	55.1114 (12)	0.3234 (9)	0.1014 (14)	0.0758 (13)	0.4065 (14)	0.2655 (12)	0.0862 (13)
DimeNet++ (kgcnn v2.1.0)	49.0243 (12)	37.4619 (6)	0.3400 (12)	0.0792 (6)	0.0572 (6)	0.0376 (8)	0.1993 (5)	0.0235 (5)
Finder_v1.2 composition only version	47.9614 (11)	46.5751 (10)	0.3204 (8)	0.0996 (13)	0.0764 (14)	0.6450 (16)	0.2308 (10)	0.0839 (12)
Finder_v1.2 structure based version	46.1339 (10)	50.7406 (11)	0.3197 (7)	0.0910 (12)	0.0693 (11)	0.0320 (4)	0.2193 (7)	0.0343 (8)
MegNet (kgcnn v2.1.0)	54.1719 (15)	28.7606 (1)	0.3391 (11)	0.0871 (9)	0.0668 (10)	0.0352 (6)	0.1934 (4)	0.0252 (6)
MODNet (v0.1.12)	33.1918 (2)	34.2751 (5)	0.2711 (1)	0.0731 (4)	0.0548 (3)	0.0908 (10)	0.2199 (8)	0.0448 (10)
MODNet (v0.1.10)	34.5368 (3)	38.7524 (7)	0.2970 (2)	0.0731 (5)	0.0548 (4)	0.0908 (11)	0.2199 (9)	0.0448 (11)
RF-SCM/Magpie	50.0440 (14)	67.6126 (15)	0.4196 (14)	0.1040 (15)	0.0820 (15)	0.2355 (13)	0.3452 (15)	0.1165 (14)
SchNet (kgcnn v2.1.0)	42.6637 (7)	38.9636 (8)	0.3277 (10)	0.0796 (7)	0.0590 (7)	0.0342 (5)	0.2352 (11)	0.0218 (4)
MGT	31.4223 (1)	39.0179 (9)	0.3047 (3)	0.0840 (8)	0.0636 (8)	0.0361 (7)	0.2145 (6)	0.0378 (9)
Dummy	67.2851 (16)	323.9822 (16)	0.8088 (16)	0.2931 (16)	0.2897 (16)	0.5660 (15)	1.3272 (16)	1.0059 (16)

ers from crystal structures, computed with the OptB88vdW[37] and TBmBJ[38] exchange-correlation DFT functionals. The "Phonons" task is dedicated to predicting the vibrational properties of crystal structures computed using the ABINIT within 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-Ernzerhor[39] (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.[40] Since molecular layers usually are connected through non-covalent weak interactions,[41–43] 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 Formation Energy task is a regression task for predicting the energy required to form the crystal structure. As the energy needed to form a covalent or ionic bond between two atoms is much greater than the electrostatic interaction between them, the bond energy plays a dominant role in determining the formation energy of a molecule or compound. Therefore, including the electrostatic interaction without using a filtering function to correctly add the processed electrostatics to the bonded interactions, and using simplified Coulomb interactions can result in a negative impact on the performance of the model.

From the test results on the remaining tasks in the MatBench dataset, the incorporation of the attention module for long-range interactions shows promising results. While the overall MAE of MGT across all tasks is higher than that of ALIGNN, it ranks fifth out of 16 models, as shown in Table S1 (Supporting Information), and it still manages to rank in the top 10 best-performing models across all tasks. This demonstrates its broad applicability and competitive performance.

The QMOF[32] consists of 20375 Metal Organic Framework (MOF) structures with results from electronic structure calculations done using the functionals: PBE,[39] High Local Exchange 2017[44] (HLE17) and Heyd–Scuseria–Ernzerhof[45] 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

Property	**MGT**	**ALIGNN**	**CGCNN**
Bandgap (eV)	0.240	0.224	0.330
HOMO (eV)	0.263	0.245	0.361
LUMO (eV)	0.252	0.232	0.330

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[32] 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.

Component	Number of Repetitions			
	1	2	3	4
MHA	0.4031	0.3981	0.3880	0.3816
ALIGNN	0.3224	0.2894	0.2767	0.2609
EGGC	0.3840	0.3328	0.3152	0.3016

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.

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.

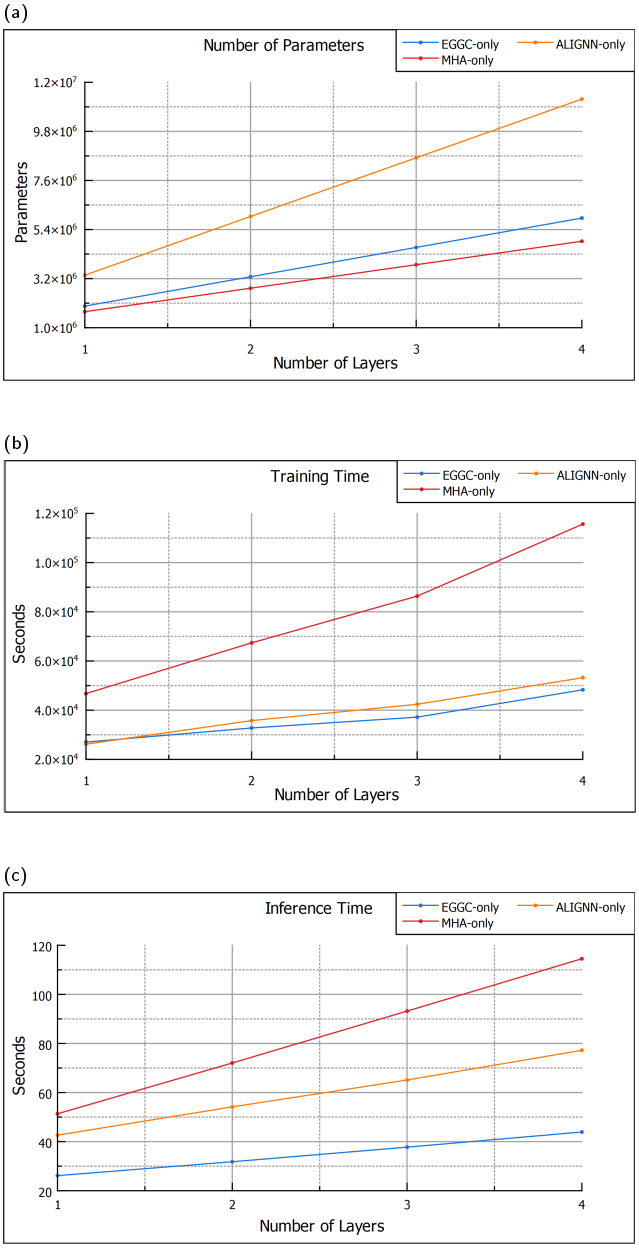


Fig. 4 Graphs showing the impact that each module within the MGT Encoder has on model size, training and inference. (a) shows the impact of the modules on the number of parameters. (b) shows the influence that each module has on the training time of the model (results obtained by training the model on 4 GPUs). (c) shows the impact that the modules have on the inference time (results obtained by running the model on 1 GPU)

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 second 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 12Å, by using a simplified representation of Coulomb interactions obtained from the Coulomb Matrix[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

1 K. Lin, L. Wang and Z. Liu, Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 12939–12948.

2 Z. Guo and H. Wang, *IEEE Transactions on Industrial Informatics*, 2020, **17**, 2776–2783.

3 Z. Chen, X. Li and J. Bruna, *arXiv preprint arXiv:1705.08415*, 2017.

4 D. Arya and M. Worring, Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, 2018, pp. 117–125.

5 J. Yang, Z. Liu, S. Xiao, C. Li, D. Lian, S. Agrawal, A. Singh, G. Sun and X. Xie, *Advances in Neural Information Processing Systems*, 2021, **34**, 28798–28810.

6 C. Yang, Z. Liu, D. Zhao, M. Sun and E. Y. Chang, IJCAI, 2015, pp. 2111–2117.

7 X. Li, Y. Zhou, N. Dvornek, M. Zhang, S. Gao, J. Zhuang, D. Scheinost, L. H. Staib, P. Ventola and J. S. Duncan, *Medical Image Analysis*, 2021, **74**, 102233.

8 I. Sarasua, J. Lee and C. Wachinger, 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), 2021, pp. 1356–1359.

9 K. M. Timmins, I. C. Van der Schaaf, I. N. Vos, Y. M. Ruigrok, B. K. Velthuis and H. J. Kuijf, *IEEE Transactions on Medical Imaging*, 2023.

10 K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko and K.-R. Müller, *The Journal of Chemical Physics*, 2018, **148**, 241722.

11 T. Xie and J. C. Grossman, *Physical review letters*, 2018, **120**, 145301.

12 C. Chen, W. Ye, Y. Zuo, C. Zheng and S. P. Ong, *Chemistry of Materials*, 2019, **31**, 3564–3572.

13 K. Choudhary and B. DeCost, *npj Computational Materials*, 2021, **7**, 185.

14 T. Hsu, T. A. Pham, N. Keilbart, S. Weitzner, J. Chapman, P. Xiao, S. R. Qiu, X. Chen and B. C. Wood, *arXiv preprint arXiv:2109.11576*, 2021.

15 S. Zhang, Y. Liu and L. Xie, *arXiv preprint arXiv:2011.07457*, 2020.

16 C. W. Park and C. Wolverton, *Physical Review Materials*, 2020, **4**, 063801.

17 J. Gasteiger, J. Groß and S. Günnemann, *arXiv preprint arXiv:2003.03123*, 2020.

18 A. Y.-T. Wang, M. S. Mahmoud, M. Czasny and A. Gurlo, *Integrating Materials and Manufacturing Innovation*, 2022, **11**, 41–56.

19 S. Kearnes, K. McCloskey, M. Berndl, V. Pande and P. Riley, *Journal of computer-aided molecular design*, 2016, **30**, 595–608.

20 J. Gasteiger, S. Giri, J. T. Margraf and S. Günnemann, *arXiv preprint arXiv:2011.14115*, 2020.

21 M. Wen, S. M. Blau, E. W. C. Spotte-Smith, S. Dwaraknath and K. A. Persson, *Chemical science*, 2021, **12**, 1858–1868.

22 J. Westermayr, M. Gastegger and P. Marquetand, *The journal of physical chemistry letters*, 2020, **11**, 3828–3834.

23 D. Jha, L. Ward, A. Paul, W.-k. Liao, A. Choudhary, C. Wolverton and A. Agrawal, *Scientific reports*, 2018, **8**, 17593.

24 N. Lubbers, J. S. Smith and K. Barros, *The Journal of chemical physics*, 2018, **148**, 241715.

25 E. Kocer, T. W. Ko and J. Behler, *Annual Review of Physical Chemistry*, 2022, **73**, 163–186.

26 O. T. Unke and M. Meuwly, *Journal of chemical theory and computation*, 2019, **15**, 3678–3693.

27 T. W. Ko, J. A. Finkler, S. Goedecker and J. Behler, *Nature communications*, 2021, **12**, 398.

28 X. Xie, K. A. Persson and D. W. Small, *Journal of Chemical Theory and Computation*, 2020, **16**, 4256–4270.

29 S. A. Ghasemi, A. Hofstetter, S. Saha and S. Goedecker, *Phys. Rev. B*, 2015, **92**, 045131.

30 M. Rupp, A. Tkatchenko, K.-R. Müller and O. A. Von Lilienfeld, *Physical review letters*, 2012, **108**, 058301.

31 A. Dunn, Q. Wang, A. Ganose, D. Dopp and A. Jain, *npj Computational Materials*, 2020, **6**, 138.

32 A. S. Rosen, V. Fung, P. Huck, C. T. O'Donnell, M. K. Horton, D. G. Truhlar, K. A. Persson, J. M. Notestein and R. Q. Snurr, *npj Computational Materials*, 2022, **8**, 112.

33 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, *Advances in neural information processing systems*, 2017, **30**, year.

34 S. Elfwing, E. Uchibe and K. Doya, *Neural Networks*, 2018, **107**, 3–11.
35 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, *Advances in neural information processing systems*, 2019, **32**, year.
36 M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai *et al.*, *arXiv preprint arXiv:1909.01315*, 2019.
37 J. Klimeš, D. R. Bowler and A. Michaelides, *Journal of Physics: Condensed Matter*, 2009, **22**, 022201.
38 F. Tran and P. Blaha, *Physical review letters*, 2009, **102**, 226401.
39 J. P. Perdew, K. Burke and M. Ernzerhof, *Physical review letters*, 1996, **77**, 3865.
40 J. H. Jung, C.-H. Park and J. Ihm, *Nano letters*, 2018, **18**, 2759–2765.
41 Y. S. Al-Hamdani and A. Tkatchenko, *The Journal of chemical physics*, 2019, **150**, year.
42 F. Haase, K. Gottschling, L. Stegbauer, L. Germann, R. Gutzler, V. Duppel, V. Vyas, K. Kern, R. Dinnebier and B. Lotsch, *Materials Chemistry Frontiers*, 2017, **1**, 1354–1361.
43 K. T. Mahmudov, M. N. Kopylovich, M. F. C. G. da Silva and A. J. Pombeiro, *Coordination Chemistry Reviews*, 2017, **345**, 54–72.
44 P. Verma and D. G. Truhlar, *The Journal of Physical Chemistry C*, 2017, **121**, 7144–7154.
45 J. Heyd, G. E. Scuseria and M. Ernzerhof, *The Journal of chemical physics*, 2003, **118**, 8207–8215.