

Expressive Graph Informer Networks

Jaak Simm(✉), Adam Arany, Edward De Brouwer, and Yves Moreau

ESAT-STADIUS, KU Leuven, 3001 Leuven, Belgium
{jaak.simm,adam.arany,edward.debrouwer,moreau}@esat.kuleuven.be

Abstract. Applying machine learning to molecules is challenging because of their natural representation as graphs rather than vectors. Several architectures have been recently proposed for deep learning from molecular graphs, but they suffer from information bottlenecks because they only pass information from a graph node to its direct neighbors. Here, we introduce a more expressive route-based multi-attention mechanism that incorporates features from routes between node pairs. We call the resulting method Graph Informer. A single network layer can therefore attend to nodes several steps away. We show empirically that the proposed method compares favorably against existing approaches in two prediction tasks: (1) 13C Nuclear Magnetic Resonance (NMR) spectra, improving the state-of-the-art with an MAE of 1.35 ppm and (2) predicting drug bioactivity and toxicity. Additionally, we develop a variant called injective Graph Informer that is provably more powerful than the Weisfeiler-Lehman test for graph isomorphism. We demonstrate that the route information allows the method to be informed about the *non-local topology* of the graph and, thus, it goes beyond the capabilities of the Weisfeiler-Lehman test. Our code is available at github.com/jaak-s/graphinformer.

Keywords: Graph neural networks · Graph attention · Self-attention · NMR · Bioactivity prediction

1 Introduction

Graphs are used as a natural representation for objects in many domains, such as compounds in computational chemistry or protein–protein interaction networks in systems biology. Machine learning approaches for graphs fall into two main lines of research: the *spectral* and the *spatial* approach. The spectral approach relies on the eigenvalue decomposition of the Laplacian of the graph and is well suited for problems involving a single fixed graph structure.

By contrast, *spatial* graph methods work directly on the nodes and edges of the graph. Convolutional neural networks (CNN) have inspired many spatial methods. Similarly to a local convolution filter running on the 2D grid of an image, spatial approaches update the hidden vectors of a graph node by aggregating the hidden vectors of its neighbors. Several spatial graph methods have

J. Simm and A. Arany—Contributed equally as first authors.

© Springer Nature Switzerland AG 2022
G. Nicosia et al. (Eds.): LOD 2021, LNCS 13164, pp. 198–212, 2022.
https://doi.org/10.1007/978-3-030-95470-3_15

been proposed, which vary in how to carry out this update. The most straightforward approach is to sum (or average) the hidden vectors of neighbors, and then transform the results with a linear layer and a nonlinearity (*e.g.*, as proposed in Neural Fingerprints [10] and Graph Convolutional Networks [19]). Instead of a simple dense feedforward layer, some methods use GRU-based gating [22], edge hidden vectors [17], and attention to the neighbors [29]. One of the advantages of spatial methods over spectral methods is that they can be straightforwardly applied to problems involving multiple graphs.

However, the update step of a node in spatial methods only has access to its own neighborhood, which limits the information flow throughout the whole graph. Increasing the accessible neighborhood of each node requires the stacking of several layers. The dilated filters in CNNs [32] are an example of solving this neighborhood limitation in images, by providing an efficient way to gather information over longer distances (see Fig. 1, (A) and (B)).

In this work, we propose a method that flexibly aggregates information over longer graph distances in one step, analogous to dilated filters. Our *Graph Informer* approach is inspired by the sequence-to-sequence transformer model [28]. The core contribution of our work is the introduction of *route-based multi-head self-attention* (RouteMHSA), which allows the attention mechanism to also access route information and, thus, base its attention scores both on the features of the nodes and the route between them. This enables Graph Informer to gather information from nodes that are not just direct neighbors. Furthermore, the proposed approach can straightforwardly use edge features, as they can be incorporated into the route features. In the case of chemical compounds, this allows using the annotation of the bonds between the atoms (single, double, aromatic, etc.). The central idea is illustrated in Fig. 1. Additionally, we investigate the expressiveness of Graph Informer, by describing a variant that is provably as powerful as the Weisfeiler-Lehman (WL) test and showing empirically that it can even go beyond the WL test.

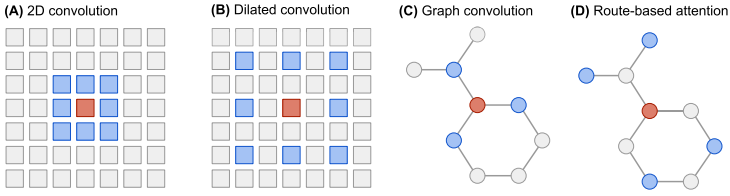


Fig. 1. Illustration of how route-based graph attention can mimic dilated filters in graphs. *Red* colored nodes are updated based on the vectors of the *blue* color nodes. (A) 2D convolutional filter, (B) 2D dilated filter, (C) graph convolution, (D) route-based graph attention. (Color figure online)

Our Graph Informer compares favorably with state-of-the-art graph-based neural networks on 13C NMR spectrum and drug bioactivity prediction tasks, which makes it suitable for both *node-level* and *graph-level* prediction tasks.

2 Proposed Approach

2.1 Setup

In this work, we consider learning problems on graphs where both nodes and edges are labeled. For a graph G with N nodes, we denote its node features by a matrix $X \in \mathbb{R}^{N \times F_{\text{nodes}}}$ and the route information by tensor $P \in \mathbb{R}^{N \times N \times F_{\text{route}}}$, F_{nodes} and F_{route} are the dimensions of the nodes and route features, respectively and the route information P is computed using the adjacency matrix of the graph and the edge labels. For example, for chemical compound graphs, where nodes are atoms, the vector between a pair of atoms in the route information tensor (*e.g.*, vector $P[i, j]$ for atoms i and j) can contain information on the type of route that connects the two atoms (rigid, flexible, aromatic) and how far they are from each other (shortest path).

We propose a method that works for both (1) *graph-level* tasks, such as graph classification where the whole graph G is assigned a label, and (2) *node-level* tasks where one should predict a value or a label for nodes (or subset of nodes) of the graph.

Due to the frequent use of tensors, in the following we adapt the Einstein summation convention, where the corresponding lower and upper indices are contracted (summed over). In case of matrices we use the matrix notation and the Einstein notation interchangeably for easier comparison to previous works, e.g. $AB = A_{ik}B^k_j$ corresponds to $\sum_k a_{ik}b_{kj}$. Note that $A_{ik}B_{jk}$ corresponds to a rank-3 tensor with elements $a_{ik}b_{jk}$ and not a matrix and that we consider the Euclidean metric tensor and hence use covariant (lower indices) and contravariant (upper indices) representations interchangeably.

2.2 Dot-Product Self-attention

Our approach is inspired by the encoder used in the transformer network [28], which proposed dot-product attention for sequence models formulated as

$$\text{Attn}(Q, K, V) = \sigma\left(\frac{1}{\sqrt{d_k}} QK^\top\right) V = \sigma\left(\frac{1}{\sqrt{d_k}} Q_{if} K^f_j\right) V^j_{f'},$$

where $Q, K \in \mathbb{R}^{N \times d_k}$ and $V \in \mathbb{R}^{N \times d_v}$ are the query, key, and value matrices, i, j are node indices, f, f' are embedding indices and σ is the Softmax function, which makes sure that the attention probabilities for each node *sum to one*. This attention can be used as *self-attention* by projecting the input d-dimensional hidden vectors $H \in \mathbb{R}^{N \times d}$ into queries, keys and values. Q, K and V are computed from the input hidden vectors with corresponding weight matrices $W_Q \in \mathbb{R}^{d_k \times d}, W_K \in \mathbb{R}^{d_k \times d}$ and $W_V \in \mathbb{R}^{d_v \times d}$ (*i.e.*, $Q = HW_Q^\top$, $K = HW_K^\top$ and $V = HW_V^\top$). Intuitively, the i-th row of Q is the query vector of the i-th node (attending node), the j-th row of K is the key vector of node j (attended node) and the j-th row of V is the value vector for node j (value of the attended node).

To encode sequences with dot-product attention, the original transformer network [28] adds positional encodings to the hidden vectors H. In particular, sine waves with different frequencies and phases were used in the original paper to encode the position of the sequence elements. This allows the network to change attention probabilities based on the positions. However, as presented in the next section, such positional encoding is not directly applicable to graphs.

2.3 Route-Based Dot-Product Self-attention

One aspect to note is that in the case of graphs, there is no analogue for the global position of nodes. Therefore, we propose a novel *stable relative addressing* mechanism for graphs. The proposed mechanism adds a new component to the dot-attention that depends on the *features of the route* between two nodes in the graph. This route component is made up of a query and a key part. For the *route query* $Q_R \in \mathbb{R}^{N \times d_r}$ we map the hidden vectors of the nodes (H) with a matrix $W_Q^{\text{route}} \in \mathbb{R}^{d_r \times d}$

$$Q_R = H(W_Q^{\text{route}})^\top,$$

where d_r is the dimension of the route query and W_Q is the query weight matrix. Similarly, for the *route key* $K_R \in \mathbb{R}^{N \times N \times d_r}$ we map the route features P with a matrix $W_K^{\text{route}} \in \mathbb{R}^{d_r \times d}$

$$(K_R)_{ijf} = P_{ijh}(W_K^{\text{route}})_f^h,$$

where i, j refer to nodes dimensions, f to the embedding dimension and h to route features. The route information allows the attention mechanism to access the topology of the graph. Note that while Q_R is a matrix of route queries, one for each node of the graph, K_R is a tensor of rank 3, with each slice $K_R[i, j]$ corresponding to the route from node i to node j, and f indexing route embedding.

The route query and key can be now contracted into a $N \times N$ matrix of (logit) attention scores, which can be added to the original node-based scores $QK^\top$.

The intuition behind this is straight-forward: we compute the scalar product between the query vector $Q_R[i]$ of the node i and the key vector $K_R[i, j]$ (by contracting the index f) of the route from i to j.

Now the route-based attention probabilities $A \in [0, 1]^{N \times N}$ are given by

$$A_{ij} = \sigma\left(\frac{1}{\sqrt{d_k + d_r}}\left(Q_{if}K_j^f + (Q_R)_{if}(K_R)_{ij}^f\right)\right), \tag{1}$$

where we also added the size of the route key d_r to the normalization for keeping the values within trainable (non-plateau) ranges of the softmax function. This enables the network to use both the information on the routes ($(Q_R)_{if}(K_R)_{ij}^f$), as well as the node hidden vectors ($Q_{if}K_j^f$), to decide the attention probabilities.

Analogously, we project the route information into the value component V of the dot-product attention. Firstly, we map the route data P into

$$(V_R)_{ijf} = P^h_{ij}(W^{\text{route}}_V)_{fh},$$

where $V_R \in \mathbb{R}^{N\times N\times d_v}$ is the tensor of rank 3 with slice $V_R[i,j]$ corresponding to the value vector for the route from node i to node j. The route value tensor V_R can then be weighted according to the attention probabilities A from Eq. 1 giving as the final route-based graph attention:

$$\text{RouteAttn}(Q,K,V,Q_R,K_R,V_R)_{if} = A_{ij}V^j_f + A_{i,j}(V_R)^j_{if}. \tag{2}$$

2.4 Locality-Constrained Attention

The route-based attention defined in Eq. 2 is general and allows unconstrained access throughout the graph. In some applications, localized access is preferred and this can be easily forced by adding a masking matrix to the attention scores:

$$(A_{\text{LC}})_{ij} = \sigma\left(\frac{1}{\sqrt{d_k+d_r}}\left(Q_{if}K^f_j + (Q_R)_{if}(K_R)^f_{ij}\right) + (M_{\text{route}})_{ij}\right), \tag{3}$$

where $M_{\text{route}} \in \{-\infty, 0\}^{N\times N}$ is 0 for unmasked routes and minus infinity for masked connections (in practice, we use a large negative value). This allows us, for example, to mask all nodes whose shortest distance from the current node is larger than 3 or some other number, thereby creating an attention ball around the current node. Similarly, it is possible to use the mask M_{route} to create an attention shell (*i.e.*, attending nodes within a given range of shortest distances).

In our experiments, we only used the attention ball with different values for the radius.

2.5 Implementation Details

As in the transformer architecture, we use multiple heads. Several route-based self-attentions are executed in parallel and their results are concatenated, allowing the network to attend to several neighbors at the same times.

Pool Node. Additionally, we introduce a *pool* node that has a different embedding vector than the graph nodes. The pool node has no edges to the graph nodes, but is unmasked by M_{route}, so that the attention mechanism can always attend and read, if required. This idea has two motivations. It allows the information to be easily shared across the graph and can be used as a "not found" answer when the attention mechanism does not find the queried node within the graph nodes.

Learnable Module. We use the attention probabilities A_{LC} for RouteAttn from Eq. 2, which takes in (1) hidden vectors of the nodes H, with size $(d, N_{\mathrm{nodes}}, N_{\mathrm{hidden}})$, (2) route information P, with size $(d, N_{\mathrm{nodes}}, N_{\mathrm{nodes}}, F_{\mathrm{route}})$, and (3) masks M_{node} and M_{route}. We use multiple heads and concatenate their results, giving us a route-based multi-head self-attention module (RouteMHSA):

$$\begin{aligned}\mathrm{RouteMHSA} &= \mathrm{Concat}(X_1, \ldots, X_{N_{\mathrm{head}}}) \\ \text{with } X_i &= \mathrm{RouteAttn}(HW_{i,Q}, HW_{i,K}, HW_{i,V}, \\ &\qquad HW^{\mathrm{route}}_{i,Q}, PW^{\mathrm{route}}_{i,K}, PW^{\mathrm{route}}_{i,V}),\end{aligned}$$

where W are the learnable parameters of the module.

The values for hidden size d, key size d_k, value size d_v, route key size d_r, and the number of heads N_{heads} can be chosen independently of each other. However, in our experiments we set d equal to $d_k N_{\mathrm{heads}}$, with $N_{\mathrm{heads}} \in \{6, 8\}$. We also use $d_v = d_k = d_r$.

Computational Complexity. The computational complexity of the RouteMHSA is quadratic with respect to the number of nodes in the graph, because the attention mechanism computes all pairwise attention probabilities. In practice, this means that the method can scale to graphs of a few hundred, maximally 1,000 nodes. This is more than enough for handling drug-like compounds, which typically have less than 100 (non-hydrogen) atoms.

3 Architecture of the Network

The architecture of Graph Informer is inspired by transformer networks [28], where the output of the multi-head attention is fed through a linear layer and added to its input, then a subsequent layer normalization [3] is applied. Additionally, a feedforward network (FFN) was applied to each hidden vector separately. However, in our graph experiments, for both node-level and graph-level tasks, we found this architecture to be quite difficult to train due to gradient flow issues.

Instead, we found that creating a residual-style network [14] with LayerNorm solved the gradient flow issue and was easy to train. The layer for this architecture can be expressed as

$$T = H + \mathrm{LayerNorm}(\mathrm{Linear}(\mathrm{RouteMHSA}(H))) \tag{4}$$

$$H' = T + \mathrm{LayerNorm}(\mathrm{FFN}(T)), \tag{5}$$

with H' being the output of the block (*i.e.*, updated hidden vectors). The architecture is depicted in Fig. 2. Graph Informer uses both neuron-level and channel-level dropout, with dropout rate 0.1.

4 Expressiveness of Graph Informer

Next, we investigate the general ability of Graph Informer to distinguish between distinct but similar graphs, also known as the graph isomorphism problem.

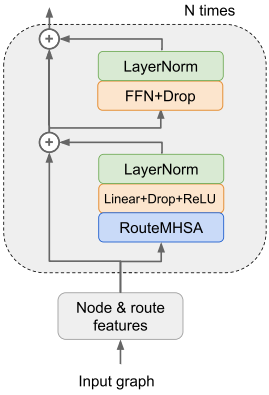


Fig. 2. Layer setup for the architectures of Graph Informer.

4.1 The Weisfeiler-Lehman Test

We propose a variant of our approach that is provably as powerful as the Weisfeiler-Lehman test of isomorphism of dimension 1 ($WL_{\mathrm{dim}=1}$). Generalizing the notation of [31] to include route information, P, we write the update function of our model as

$$h_v^k = \phi\left(h_v^{k-1}, \left\{(h_u^{k-1}, P_{v,u}) : u \in \mathcal{N}(v)\right\}\right),$$

where $\left\{(h_u^{k-1}, P_{v,u}) : u \in \mathcal{N}(v)\right\}$ is the multiset of node vectors, h_u^{k-1}, and their route information vectors $P_{v,u}$, which can be attended by node v.

Such a network is provably as powerful as the Weisfeiler-Lehman test of isomorphism if the function $\phi(.)$ is injective[1] with respect to the hidden vectors h_v^{k-1} and h_u^{k-1} arguments, as follows straightforwardly from the theorem in [31].

RouteMHSA satisfies this condition with respect to the first argument (h_v^{k-1}) because one of the heads can attend to itself. However, to make it *provably* injective with respect to the second argument, we can replace the softmax mapping in Eq. 3 by an elementwise sigmoid. We call this version the *Injective* Graph Informer.

4.2 Beyond the Weisfeiler-Lehman Test

Several graph neural networks, including GIN [31] and NeuralFP [10], are limited in their expressiveness to the Weisfeiler-Lehman test of dim = 1. By contrast, the Graph Informer can aggregate route information and, therefore, has the capability to go beyond $WL_{\mathrm{dim}=1}$ as the routes between the nodes reveal non-local topology. Indeed, we can show that Graph Informer has the following capability:

[1] We assume here that the output function is also injective.

Lemma 1. *Injective Graph Informer using the histogram of the lengths of all (possibly cyclic) routes between two nodes up to length N can distinguish any pair of graphs with different spectra.*

The proof of this lemma follows from the fact that injective Graph Informer is injective with respect to route features P and that graphs are cospectral (in terms of their adjacency matrix) if and only if $\mathrm{Tr}(A^r)$ are equal for both matrices for all values of r [2]. Notably, our method offers a more detailed view because it can also distinguish cospectral graphs (*e.g.*, the Tesseract (Q4) and Hoffman graphs, which are both 16-node and 4-regular).

We hypothesize that in practice this capability of Graph Informer gives strong additional expressiveness and makes it difficult to construct counterexamples because the route information allows the method to capture the non-local topology of the graph in detail.

5 Related Research

There are several neural network approaches that can directly work on the graph inputs. The key component in many of the works is to propagate hidden vectors of the nodes H using the adjacency matrix[2] A, either by summing the neighbor vectors, AH, or by averaging them as $\tilde{A}H$, where $\tilde{A} = A(D)^{-1}$ is the normalized adjacency matrix with D being the diagonal matrix of degrees. Neural Fingerprint [10] proposed to use the propagation AH, combined with a node-wise linear layer and a nonlinearity to compute fingerprints for chemical compounds. Based on the degree of the node, Neural Fingerprint uses a different set of linear weights. Similarly, Graph Convolutional Networks (GCN, [19]) propose to combine AH with a linear layer and a nonlinearity for semi-supervised learning, but by contrast with Neural Fingerprint, they use the same linear layer for all nodes, irrespective of their degree. Kearnes *et al.* [17] proposed the Weave architecture, where they use embedding vectors also for the edges of the graph and the propagation step updates nodes and edges in a coupled way.

In a separate line of research, [25] proposed Graph Neural Networks (GNN), which also use AH to propagate the hidden vectors between the nodes, but instead of passing the output to the next layer, the same layer is executed until convergence, which is known as the Almeida-Pineda algorithm [1]. A recent GNN-based approach is called Gated Graph Neural Networks (GGNN) [22], where at each iteration the neighborhood information AH is fed into a GRU [7]. An iterative graph network approach, motivated by the Message Passing algorithm in graphical model inference [24] was proposed by [8]. Gilmer et al. [11] generalized the existing approaches to the Neural Message Passing (NeuralMP) algorithm and showed the effectiveness of the approach for predicting simulated quantum properties based on 3D structures of compounds.

Hamilton et al. [13] propose a representation learning method, GraphSAGE, that does not require supervised signals and inductively embeds nodes in large graphs.

[2] The adjacency matrix here is assumed to include the self-connection (*i.e.*, $A_{ii} = 1$).

The method closest to ours is Graph Attention Networks (GAT) [29]. Similarly to us, they proposed to use multi-head attention to transform the hidden vectors of the graph nodes. However, GAT cannot integrate route features and its attention is only limited to the neighbors of every node. For graph isomorphism testing, this means it fails to separate any regular graphs, similar to $WL_{\mathrm{dim}=1}$.

In contrast to the spatial methods reviewed above, spectral methods use the graph Fourier basis. One of the first spectral approaches to introduce this concept in the context of neural networks was SCNN [6]. To avoid the costly eigenvalue decomposition, Chebyshev polynomials were proposed to define localized filters [9]. Based on this work, proposed a neural network that combines spectral methods with metric learning for graph classification was proposed [20].

For sake of completeness, we note that another loosely related line of research has been working on the problem of transforming an input graph to a target graph and are known as *graph transformers* [5,33]. Those works usually do not use the attention mechanism used in the transformer network [28]. Yet, in a recent communication, also called graph transformers, [21] proposed a node-level attention mechanism for the same graph-to-graph task. In contrast, our model tackles node- or graph-level classification and regression tasks, and uses route-level attention mechanism.

In graph isomorphism research, as already mentioned, [31] showed that injective update rule and readout are crucial to guarantee expressiveness at the level of $WL_{\mathrm{dim}=1}$. In a recent publication, [23] proposed a graph neural network as powerful as $WL_{\mathrm{dim}=3}$.

In one of our main application areas, Nuclear Magnetic Resonance (NMR) chemical shift prediction (covered in Sect. 6.2), is an important element in structure elucidation. Recently, [15] proposed an imitation learning approach to solve this NMR inverse problem.

6 Evaluation

We compare our method against several baselines on two tasks. We first consider a *node-level* task where, given the graph of the compound, the goal is to predict 13C NMR peaks for all of the carbon atoms, see Sect. 6.2. For the *graph-level* tasks, we consider the drug–protein activity data set ChEMBL [4] and compound toxicity prediction from MoleculeNet [30], see Sect. 6.3. In all tasks and for all methods, we used the same feature set (see Appendix). For methods that could not use route features but supported edge features, we used the bond type (single, double, triple, aromatic). We consider an extensive set of baselines: GGNN [22], NeuralFP [10], GCN [19], Weave [17], GAT [29], NeuralMP [11] and GIN [31].

6.1 Model Selection

All methods used the validation set to choose the optimal hidden size from $\{96, 144, 192, 384\}$ and dropout from $\{0.0, 0.1\}$ [27]. For our method, attention radius from $\{1, 2, 3, 4\}$ and head count from $\{6, 8\}$ were also selected. All methods

were trained with the Adam optimizer [18]. The validation set was used for early stopping (*i.e.*, we executed all methods for 100 epochs and picked the best model). At epochs 40 and 70, the learning rate was decreased by a factor of 0.3. Finally, the best early stopped model was used to measure the performance on the test set.

6.2 Node-Level Task: NMR 13C Spectrum Prediction

13C NMR is a cost-efficient spectroscopic technique in organic chemistry for molecular structure identification and structure elucidation. If 13C NMR measurements can be combined with a highly accurate prediction, based on the graph (structure) of the molecule, it is possible to validate the structure of the sample. It is crucial to have highly accurate predictions, because in many cases several atoms have peaks that are very close to each other. If such predictions are available, the chemist can avoid running additional expensive and time-consuming experiments for structure validation.

To create the data set, we used 13C NMR spectra from NMRShiftDB2.[3] After basic quality filtering 25,645 molecules remained with peak positions raging from –10 to 250 ppm (parts per million) with mean and standard deviation of 95 and 52 ppm.

In accordance to the most common Lorentzian peak shape assumption in NMR spectroscopy, we minimize the Mean Absolute Error (MAE) [12]. For evaluation, we split the data into train-validation-test sets containing 20,000, 2,500, and 3,145 molecules, respectively, and report results over 3 repeats with different splits. All methods use the same node-level output head, consisting of a tanh layer followed by a linear output layer. The MAE loss is minimized only for the atoms that have measurements for the 13C peaks (*i.e.*, the carbon atoms). For the results, see Table 1.

Our proposed Graph Informer reaches 1.35 MAE, which is also better than the state-of-the-art results from computational chemistry literature, which have reported 1.43 MAE for 13C for the NMRShiftDB2 [16]. At these levels of accuracy, many molecules with *densely packed* 13C NMR spectra can be resolved. This class of molecules with *densely packed* spectra are common for aromatic ring systems, which are prevalent in drug-like compounds. As an example, see Fig. 3a depicting 2-chloro-4-fluorophenol, which has three carbons, labelled 1, 3, 4 with peaks within the 115.2 to 116.6 ppm range. Figure 3b displays the predictions from the molecule when it was in the test set. The predictions match closely the true spectrum and, critically, predict the true ordering, which allows us to perfectly match peaks.

For a Graph Informer with 3 layers and 6 heads all having radius 2, we depict the attention probabilities in Fig. 4 for the bottom carbon (number 1 in Fig. 3a) of same 2-chloro-4-fluorophenol compound. We can see that all heads in Layer 1 strongly use the pool node as a reference. Also, we can see that in Layer 1 the

[3] Available here: https://nmrshiftdb.nmr.uni-koeln.de/.

Table 1. Test MAEs for NMR 13C spectrum prediction averaged over different folds.

Method	#layers	Test MAE
Mean model		46.894
GCN	2	6.284 ± 0.053
GCN	3	8.195 ± 0.385
GIN	2	5.015 ± 0.082
GIN	3	6.275 ± 0.306
GGNN	*na*	1.726 ± 0.055
NeuralMP	*na*	1.704 ± 0.025
GAT	1	2.889 ± 0.021
GAT	2	3.148 ± 0.067
GAT	3	5.193 ± 0.196
NeuralFP	1	3.839 ± 0.039
NeuralFP	2	2.248 ± 0.027
NeuralFP	3	2.040 ± 0.037
NeuralFP	4	1.966 ± 0.045
Graph Informer	1	1.827 ± 0.035
Graph Informer	2	1.427 ± 0.024
Graph Informer	3	1.375 ± 0.016
Graph Informer	4	**1.348** ± 0.007
State of the art [16]		1.43 ± NA

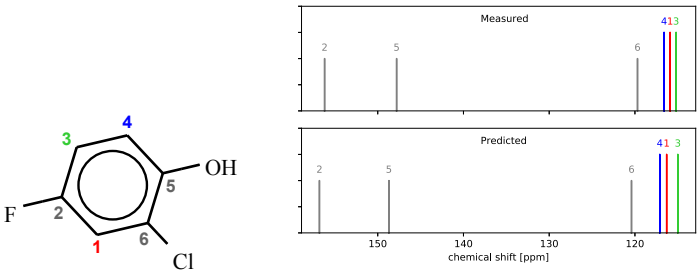
Table 2. AUC-ROC scores for multi-task datasets ChEMBL and Tox21 of MoleculeNet averaged over different folds.

Method	ChEMBL	Tox21
NeuralFP	0.826 ± 0.005	0.829
GCN	0.814 ± 0.005	0.809
GIN	0.816 ± 0.004	0.810
GGNN	0.756 ± 0.014	0.809
NeuralMP	0.707 ± 0.005	0.804
Weave	0.830 ± 0.006	0.822
Graph Informer	**0.839** ± 0.003	**0.848**

first head focuses on the atom itself, the second head attends to all neighbors within radius 2, and the third head focuses on the halogen atoms.

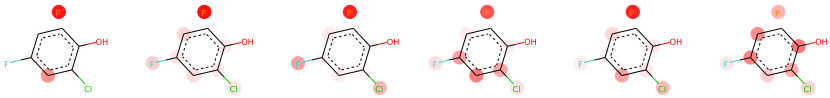
6.3 Results for Graph-Level Tasks

We benchmark the methods on two drug bioactivity data sets: ChEMBL v23 by [4] (94 tasks, 87,415 molecules, and 349,192 activity points) and Tox21 from MoleculeNet by [30] (12 tasks, 7,831 molecules, and 77,946 activity points). For ChEMBL, we used a repeated holdout-validation, and for Tox21 we used the single train-validation-test split given by MoleculeNet. In the case of ChEMBL, because of well-known series effect in drug design, allocating molecules randomly to the training and test sets results in overly optimistic and inconsistent performance estimates [26] because highly similar molecules end up in both training and test sets. Therefore, we apply *clustered* hold-out validation. We allocate the clusters (rather than individual molecules) to the training (60%), validation (20%), and test sets (20%). We repeat this split 5 times.



(a) Molecule structure (b) Measured and predicted spectra

Fig. 3. 13C NMR spectrum for 2-chloro-4-fluorophenol. The predicted NMR peaks are in the same order, even in the densely packed region (carbons 4, 1, 3).



(a) L1H1 (b) L1H2 (c) L1H3 (d) L2H1 (e) L2H2 (f) L2H3

Fig. 4. Attention probabilities from carbon 1 (bottom in the figures) in the 2-chloro-4-fluorophenol molecule. We show selected three heads (H) from layer 1 and layer 2 (L).

Our and all adapted methods (GCN, GIN and GGNN) use a graph pooling layer consisting of a linear layer, ReLU, mean pooling, and a linear layer. We minimize the cross-entropy loss and Table 2 reports the AUC-ROC averages over tasks for each data set. Graph Informer outperforms all baseline methods in both ChEMBL and Tox21 data sets.

7 Conclusion

In this work, we proposed a route-based multi-head attention mechanism that allows the attention to use relative position and the type of connection between the pair of nodes in the graph. Because of its route features, our approach can incorporate information from non-direct neighbors efficiently—similar to the use of dilated filters in CNNs. In our theoretical analysis, we showed that a variant of Graph Informer is provably as powerful as the $WL_{\dim=1}$ test. Our empirical evaluation demonstrated that the proposed method is suitable both for node-level and graph-level prediction tasks, and delivers significant improvements over existing approaches in 13C NMR spectrum and drug bioactivity prediction.

Acknowledgments. YM is funded by Research Council KU Leuven: C14/18/092 SymBioSys3, C3/20/100 and CELSA/17/032; CELSA-HIDUCTION CELSA/17/032; FWO (Elixir Belgium I002819N and Infrastructure I002919N); Flemish AI Research Program; Authors are affiliated to Leuven.AI - KU Leuven institute for AI; VLAIO PM (HBC.2019.2528) and MaDeSMart (HBC.2018.2287); IMI initiative MELLODDY grant No 831472; EU H2020 grant No. 956832. EDB is funded by the FWO.

Appendix

Node and Route Features

In addition to the following features, we fed the one-hot encoding of the atom types by concatenating it to the node feature vector, see Table 3. The atom types that occurred in the two data sets are {C, N, O, Cl, F, S, I, Br, P, B, Zn, Si, Li, Na, Mg, K}. As the route features (Table 4) contain all edge labels (single, double, ...), all information about the graph topology is retained.

Table 3. Node features

Position	Description
0–2	Formal charge, one-hot encoded {−1, 0, +1}
3–7	Hybridization state, one-hot encoded {s, sp, sp2, sp3}
8–13	Explicit valence, one-hot encoded integer, between 0 and 5
14	Aromaticity, binary
15	Whether it is in a ring size 3, binary
16	Whether it is in a ring size 4, binary
17	Whether it is in a ring size 5, binary
18	Whether it is in a ring size 6, binary
19	Whether it is in any ring, binary
20	Partial charge, computed by Gasteiger method, real number
21	Whether it is a H-acceptor, binary
22	Whether it is a H-donor, binary
23	Whether it is an R stereo center, binary
22	Whether it is an S stereo center, binary

Table 4. Route features

Position	Description
0–8	Bond distance, binned, [0, 1, 2, 3, 4, 5 ≤ d ≤ 6, 7 ≤ d ≤ 8, 9 ≤ d ≤ 12, 13 ≤ d]
9	Whether the shortest pure conjugated path containing at most 4 bonds, binary
10	Whether the shortest pure conjugated path containing at least 5 bonds, binary
11	Whether there is a route containing only up to 13 single bonds, binary
12	Whether there is a route containing only up to 13 double bonds, binary
13	Triple bond, binary
14	Whether there is a shortest path which is a pure conjugated path, binary
15	Whether the endpoints are in the same ring, binary
16	Single bond, binary
17	Double bond, binary
18	Aromatic bond, binary

References

1. Almeida, L.B.: A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In: Proceedings, 1st First International Conference on Neural Networks, vol. 2, pp. 609–618. IEEE (1987)
2. Alzaga, A., Iglesias, R., Pignol, R.: Spectra of symmetric powers of graphs and the Weisfeiler-Lehman refinements. arXiv preprint arXiv:0801.2322 (2008)
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 [cs, stat], July 2016
4. Bento, A.P., et al.: The ChEMBL bioactivity database: an update. Nucleic Acids Res. **42**(D1), D1083–D1090 (2014)
5. Bottou, L., Bengio, Y., Le Cun, Y.: Global training of document processing systems using graph transformer networks. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 489–494. IEEE (1997)
6. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
7. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR arXiv:1406.1078 (2014)
8. Dai, H., Dai, B., Song, L.: Discriminative embeddings of latent variable models for structured data. In: International Conference on Machine Learning, pp. 2702–2711 (2016)
9. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844–3852 (2016)
10. Duvenaud, D.K., et al.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in Neural Information Processing Systems, pp. 2224–2232 (2015)
11. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1263–1272. JMLR. org (2017)

12. Gunther, H.: NMR Spectroscopy: An Introduction, vol. 81. Wiley, Hoboken (1980)
13. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
15. Jonas, E.: Deep imitation learning for molecular inverse problems. In: Advances in Neural Information Processing Systems, pp. 4991–5001 (2019)
16. Jonas, E., Kuhn, S.: Rapid prediction of NMR spectral properties with quantified uncertainty. J. Cheminform. **11**(1), 1–7 (2019)
17. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. J. Comput. Aided Mol. Des. **30**(8), 595–608 (2016). https://doi.org/10.1007/s10822-016-9938-8
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
20. Li, R., Wang, S., Zhu, F., Huang, J.: Adaptive graph convolutional neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
21. Li, Y., Liang, X., Hu, Z., Chen, Y., Xing, E.P.: Graph transformer (2019). https://openreview.net/forum?id=HJei-2RcK7
22. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)
23. Maron, H., Ben-Hamu, H., Serviansky, H., Lipman, Y.: Provably powerful graph networks. In: Advances in Neural Information Processing Systems, pp. 2153–2164 (2019)
24. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Elsevier, Amsterdam (2014)
25. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)
26. Simm, J., et al.: Repurposing high-throughput image assays enables biological activity prediction for drug discovery. Cell Chem. Biol. **25**(5), 611–618 (2018)
27. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
28. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
29. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
30. Wu, Z., et al.: MoleculeNet: a benchmark for molecular machine learning. Chem. Sci. **9**(2), 513–530 (2018)
31. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019 (2019). https://openreview.net/forum?id=ryGs6iA5Km
32. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv:1511.07122 [cs], November 2015
33. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. In: Advances in Neural Information Processing Systems, pp. 11960–11970 (2019)