IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 May 2025, accepted 16 July 2025, date of publication 23 July 2025, date of current version 29 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3591966

RESEARCH ARTICLE

Graphormer Boosted: Molecular Property Prediction With Enhanced Graph Spatial and Edge Encodings

SARAH A. FADLALLAH, FRANCESC SERRATOSA, AND CARME JULIÀ
Department of Enginyeria Informàtica i Matemàtiques, Computer Engineering and Mathematics, Universitat Rovira i Virgili, 43007 Tarragona, Spain

Corresponding author: Sarah A. Fadlallah (sarah.fadlallah@urv.cat)

This work was supported in part by Martí Franquès Grant Program Programa Martí Franquès (PMF)-Personal Investigador Predoctoral en Formació (PIPF) 2021 Standard Edition, in addition to the Agencia de Gestión de Ayudas Universitarias y de Investigación (AGAUR) Research Group: ''ASCLEPIUS: Smart Technology for Smart Healthcare'' under Grant 2021SGR-00111; and in part by the Ministerio de Ciencia e Innovación (MCIN)/Agencia Estatal de Investigación (AEI)/10.13039/501100011033/Fondo Europeo de Desarrollo Regional (FEDER), Unió Europea (UE), under Project PID2022-138327OB-I00.

ABSTRACT Transformer-based architectures have gained popularity across various domains, including graph representation learning. However, selecting an optimal transformer configuration remains challenging, as attention-based models are highly sensitive to parameter choices and input value ranges. Even state-of-the-art architectures can underperform without proper tuning, while simple yet thoughtful modifications can unlock significant performance gains by better leveraging graph structures. In this work, we propose an enhancement to the Graphormer architecture that refines the attention mechanism by unifying spatial encodings, edge encodings, and similarity matrices. Specifically, we introduce nonlinear transformations, such as *softmax*, *sigmoid*, or *tanh*, to normalize these encodings within the attention calculation. This ensures balanced contributions from all terms, mitigating drawbacks caused by disparate value ranges. We explore two main approaches: 1) element-wise application of nonlinearities to bound spatial and edge encodings and 2) row-wise *softmax* normalization to emphasize their relative importance within the graph structure. The latter preserves relational information, enhancing the expressiveness of the model and improving the prioritization of node and edge. Experiments on molecular datasets demonstrate consistent performance improvements over the baseline Graphormer, highlighting the effectiveness of our approach. Additionally, our model outperforms traditional graph learning models. Our findings suggest that carefully designed nonlinear transformations over structural encodings significantly boost transformer-based graph models, offering a simple yet powerful strategy for improved graph representation learning.

INDEX TERMS Graph neural network, graph transformers, positional encoding, nonlinear functions, softmax, normalization, graph structure learning, molecular graphs.

I. INTRODUCTION

In the context of *in silico* drug discovery, graph-based molecular representations have proven highly effective for downstream tasks such as property prediction, drug-target interactions, and molecular docking [1]. These molecular embeddings can be derived from linear notations such as SMILES [2], which encode molecules as strings, or more richly from molecular graphs that capture the structural connectivity between atoms. In Machine Learning on the other hand, Graph Neural Networks (GNNs) [3] have unlocked new possibilities to leverage the rich features inherent in graph data. Variants such as Graph Convolutional Networks (GCNs) [4] and their branching implementations [5], as well as Gated Recurrent Networks (GRNs) and Graph Autoencoders (GAEs), have achieved remarkable success in tasks involving node classification, link prediction, and graph prediction and generation [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Ajit Khosla.

Despite the impressive ability offered by GNNs to capture underlying patterns in graphs, these methods face challenges that can result in information loss during the learning process. One key limitation is their difficulty in effectively handling heterogeneous graphs [7]. Since most GNNs function as low-pass filter networks, they mainly aggregate information from neighboring nodes [8]. This property causes GNN-based methods, such as Message Passing Neural Networks (MPNNs) and GCNs, to lose expressive power when applied to sparse graphs. Specifically, MPNNs are often constrained by the first-order Weisfeiler-Lehman (1-WL) test, limiting their ability to distinguish non-isomorphic structures [9]. Another prominent problem, known as oversmoothing, which occurs when the receptive field of the graph expands with the repeated aggregation of nodes, leading to the representation of all nodes becoming nearly identical [10], [11]. Furthermore, GCNs often suffer from vanishing gradients, restricting them to shallow architectures [12].

These challenges have motivated extensive efforts to improve how GNNs utilize graph structures and their associated information. Recent frameworks have been introduced to address these limitations. By combining classical MPNNs with novel architectures, researchers have mitigated many of these limitations [13]. An important innovation in this area is the integration of the attention mechanism [14]. Initially popularized in the sequence-to-sequence models known as Large Language Models (LLMs), attention mechanisms have since been applied across a wide range of domains beyond their original applications in natural language processing (NLP) tasks. In addition to text translation and generation [15], the attention mechanism has been adopted for image classification [16], audio processing [17], multimodal learning [18], [19], visual reasoning [20], and molecular property prediction tasks [21], [22]. They have also proven effective in pattern recognition problems, including action recognition [23] and handwritten text recognition [24].

This progress has also been extended to graph data. The Graph Attention Network (GAT) [25] was a pioneering framework that introduced the graph attention mechanism. Subsequently, Graph Transformer Networks [26] combined attention mechanisms with positional encoding (PE) to capture structural information while aggregating features from a wider range of nodes. This approach has significantly enhanced graph embeddings and advanced the state of the art in graph representation learning. The combination of self-attention and the inductive bias introduced by the PEs allows graph transformers to effectively aggregate information from each node. This enables the model to handle heterogeneous graphs with different types of nodes and edges, while also mitigating the oversmoothing problem [27].

The ability of self-attention to capture both local and global dependencies in graph structures is crucial, not only for enhancing model expressiveness and addressing limitations like oversquashing, but also for improving downstream tasks. Although transformer models, aided by attention mechanisms, have successfully overcome the range limitations of classical GNNs and Recurrent Neural Networks (RNNs) [28], [29], several challenges that limit their full potential still remain. Tasks like graph matching and link prediction require additional consideration when adapting attention mechanisms to ensure the proper incorporation of graph features and their underlying patterns. Since information about the graph structure, represented in edges and other descriptors, can be of valuable aid in the learning process. As such, the design of attention modules tailored to graphs and their properties remains an active and evolving area of research [30].

A. CONTRIBUTION

This work focuses on effectively integrating structural features, as the choice and representation of structural encodings depend heavily on data characteristics. Our contribution is based on the incorporation of nonlinear functions in the encodings that serve as positional information in the Graphormer architecture by offering the following:

- A thorough analysis of the Graphormer model and the effect of the different configurations of the information used as spatial and edge encodings.
- The analysis of various nonlinear functions and their effect on PEs, studying their impact on the representation quality and overall effectiveness of the Graphormer model.
- We find that the adoption of *softmax* boosts the role of the spatial and edge encodings that are added at the attention calculation step. This is reflected in the improvement of the final predictions of the model.
  This approach enhances the model's robustness and adaptability for graph-based tasks, particularly in molecular property prediction. The novelty of this approach is that it exploits these graph features without the need to add computationally expensive components to the network,.e.g., learning them with a separate architecture or enlarging the embedding dimension by concatenation.

The rest of the paper is organized as follows: Section II gives an overview of the application of attention and introduces different nonlinearities; Section III explains the proposed method; Section IV offers an analysis of the used encodings and their role in the overall performance of the model; Section V elaborates on the challenges and shortcomings of this work; and finally, Section VI concludes the paper and highlights future directions.

II. RELATED WORK

A. POSITIONAL ENCODING IN ATTENTION

Self-attention can be learned given a cosine similarity function denoted by weights representing the matrices of query (Q), key (K), and value (V) as follows:

$$A_{attn}(Q, K, V) = softmax\,(A)\,V. \tag{1}$$

In the above equation, the matrix A_{attn} represents the calculated attention applied to the values (V). In classical attention [14], the order of a token in relation to others in an input sequence, e.g., the position of a word in a sentence, is given by an encoding added to the embedding, providing relative positional information. This PE could be indexed using the sine and cosine functions, which offers a distinct position representation of each token. An absolute sinusoidal position *pos* of a token is encoded as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \tag{2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \tag{3}$$

where,

- *pos* is the token position in the sequence.
- i is the dimension index.
- d is the embedding dimension.

Before performing attention calculations, transformer models first embed each token into a vector and add its positional encoding to the representation:

$$\boldsymbol{x}' = \boldsymbol{x} + pos. \tag{4}$$

This embedding is then processed through multiple attention heads, giving rise to the term multiheaded attention (MHA). The outputs from these heads are aggregated, normalized, and then passed through a feedforward network defined as:

$$FFN(\boldsymbol{x}) = max(0, \boldsymbol{x}\boldsymbol{W_1} + b_1)\boldsymbol{W_2} + b_2. \tag{5}$$

The previous steps describes the encoder module of a transformer. The output is then aggregated, normalized, and fed into the decoder module, which also consists of stacked MHA blocks. Finally, the output undergoes a linear projection and is processed using a *softmax* function to generate the final token probabilities. More details on the architecture can be found in [14].

B. GRAPH REPRESENTATION LEARNING WITH TRANSFORMERS

The success of transformers in NLP has also inspired their adaptation to other types of data. Researchers have explored two major directions; augmenting graph neural networks (GNNs) with knowledge from pretrained LLMs using text-attributed or text-paired graphs [31], and directly applying self-attention to learn graph representations. The latter approach benefits from the global receptive field of self-attention, allowing each node to attend to every other node—overcoming common GNN limitations such as oversmoothing, oversquashing, and restricted expressiveness [25], [32].

Nonetheless, transformer-based models in general are not without limitations. One known issue is the ''softmax bottleneck''; where the model's expressiveness is constrained by the low-rank nature of softmax-attention outputs. To address this, [33] propose using mixtures of softmax functions to better approximate complex distributions without increasing embedding dimensionality.

Further refinements have emerged in the vision domain. For instance, [34] argue that adding positional encodings directly to token embeddings can dilute their effectiveness, since the two carry fundamentally different types of information. They propose concatenating positional vectors instead, preserving positional distinctiveness throughout the model. While compelling, transferring such techniques to graphs, where structure is irregular and discrete, requires additional innovation. Developing expressive, structure-aware encoding methods for graph transformers remains an open challenge. This is due to the fact that standard transformers assume sequence order, which is absent in graphs. As opposed to sequences or grids, nodes lack a regular Euclidean structure. Thus making traditional PE methods less effective for learning directly from attributed graphs. This irregularity complicates the representation of both topology and node/edge attributes. To overcome this, [35] incorporates Laplacian eigenvectors as absolute positional encodings, and enrich this with random walk encodings and hierarchical clustering embeddings as suggested by [36]. These encodings are concatenated and passed through a linear projection layer:

$$PE(\boldsymbol{v_i}) = W_{PE}[\lambda_i^{(1)}, \ldots, \lambda_i^{(k)}, \boldsymbol{RW_i}, \boldsymbol{HCE_i}], \tag{6}$$

where $\lambda_i^{(k)}$ are the top-k Laplacian eigenvectors for node v_i, and W_{PE} is a learned projection matrix.

Depending on the task, whether node classification, edge prediction, or graph-level inference, different structural aspects must be emphasized. For higher-order graph information, architectures like Simplicial Complex Neural Networks (SCNNs) extend GNNs to model multi-way relationships beyond simple pairwise edges [37]. Similarly, graph transformers have been adapted to incorporate topological features either directly into node embeddings [35] or as learnable biases applied during the attention score computation [13]. A common method to learn these topological properties is using a graph spectra. Spectral methods like Laplacian embeddings use the eigenvectors of the graph Laplacian to encode global structure [38]. Building on this idea, Spectral Attention Networks (SAN) [32] integrate spectral information as positional encodings into transformers, enabling them to capture complex graph topologies. Similarly, methods like Graph Transformer (GT) [35] and its refinement, LSPE [39], incorporate Laplacian eigenvectors to guide attention. Other hybrid models combine attention with graph convolutional networks (GCNs), using meta-paths to capture richer structural semantics [26].

1) THE GRAPHORMER MODEL

The Graphormer [40] is a transformer-based model specifically designed for graph representation learning. Unlike traditional GNNs, which rely on message passing, the Graphormer exploits the self-attention mechanism to capture local and

global structural information efficiently. Additionally, instead of assigning positions based on linear order as in LLMs, Graphormer incorporates graph-specific features. It induces the shortest path distances between nodes, and the edge features, learning their embeddings as added bias, unlike spectral methods such as SAN, which incorporate PEs only during initialization, and remain static throughout training. Another advantage of this bias over methods like the SAN and LSPE, is that their reliance on Laplacian eigenvectors limits their generalizations over graphs with highly irregular spectra. Thus, while these encodings may offer valuable information, their effect may not be fully witnessed when simply adding them as previously mentioned. This was addressed in [36] by incorporating additional encodings, e.g., Random Walk and Hierarchical Cluster Encoding, to support the structural learning of the graph. The issue with Random Walk however, is that it cannot detect isomorphic graph structures [36]. We argue that even if it is possible to inject more graph features into the network to learn a better representation, it could be worthwhile to enhance current encodings by taking better advantage of them without adding overhead costs to the model.

2) NODE EMBEDDING AND POSITIONAL ENCODING

In Graphormer, self-attention was modified to be more suitable for graph representation by incorporating additional encodings to exploit the structural information of the graph. Specifically, node centrality, graph connectivity, and edge features were used to augment the attention score. These descriptors enrich the model and allow it to learn a more complex representation depending on the use case. Whereas the node centrality was directly embedded within the node feature representation, connectivity, and edge encodings were later added to the similarity score as follows:

$$A_{ij} = \frac{\left(h_i W_Q\right)\left(h_j W_K\right)^\top}{\sqrt{d}} + b_{\phi(v_i,v_j)} + c_{ij}, \tag{7}$$

where,

- A_{ij} is an element of matrix A in 1.
- h_i denotes the encoded feature of node v_i.
- W_Q and W_K are the embedding weight matrices associated with the queries (Q) and the keys (K).
- $\top$ is the transpose of a matrix.
- d is the dimension of the attention head used to scale the dot product.
- $b_{\phi(v_i,v_j)}$ is a learned scalar identified by a function $\phi(v_i, v_j)$ that measures the spatial relation given by the shortest distance of the path between nodes (v_i, v_j).
- c_{ij} is the edge encoding obtained from the averaged dot product between the edge feature x_{e_n} and the edge embedding weights w_n^E. The feature x_{e_n}, is the n^{th} edge on the shortest path between nodes (v_i, v_j) consisting of N edges:

$$c_{ij} = \frac{1}{N}\sum_{n=1}^{N} x_{e_n}\left(w_n^E\right)^\top. \tag{8}$$

This formula extends the standard transformer attention mechanism by incorporating graph-specific structural information $b_{\phi(v_i,v_j)}$ and edge features c_{ij}. The shortest path distance is computed from the adjacency matrix using the Floyd–Warshall algorithm [41], and its encoding is learned during the model training phase. This additive formulation ensures that both feature similarity and structural proximity are considered in attention computation. Importantly, $b_{\phi(v_i,v_j)}$ introduces asymmetry in the score matrix A, which is critical for encoding directed or asymmetric molecular motifs, a property not handled by vanilla transformer attention. These modifications enable the Graphormer to effectively model graph-structured data while retaining the expressive power of transformers. The final attention for node i is computed as:

$$A_{attn}(h_i) = \sum_{j\in N_{(i)}} softmax(A_{ij}) \cdot (h_j W_V). \tag{9}$$

The training graphs are then passed into components similar to the original attention model. The difference is that the PEs are injected after the similarity score computation, while in the original architecture, the PEs are added to the token embeddings prior to the projection into query, key, and value matrices. The positional bias embedding in the Graphormer is learned during the training process, and the final token represents the given graph which could then be used for mainstream tasks.

C. NONLINEAR FUNCTIONS IN NEURAL NETWORKS

Activation functions are an essential component in neural networks. They are used to map the output of a given network or a layer. They introduce nonlinearity into the model that extends a linear model and provides a new representation of input x [42]. With this parameter, especially if the nonlinear function is differentiable, back-propagation on feedforward networks is possible, meaning that the network can be trained with gradient-based optimization.

In the case of learning representations, the model could better generalize the relations between the input vectors and the output scalar they are representing. Those nonlinearities or activation functions can be placed in the final layer of a neural network to obtain a label in classification tasks or a real value for regression tasks. They could also be used as hidden units within the inner layers of a network. In some cases, normalizing the weights can improve the optimization process of a model and accelerate convergence [43]. It allows the data to flow, helping the neural network to learn complex hierarchical patterns, while preventing multi-layer networks from collapsing into a single-layer linear model. In other words, with normalization and proper activation functions, neural networks can achieve high performance in many benchmarking tasks [44].

The *softmax* function for example, can be used to normalize the output of a neural network or a given layer. It takes a vector x as input and gives an output representing a normalized input or a probability distribution representing a class or a label. This distribution, which has values in [0, 1],

is proportional to the exponential of the input values [42], which can be calculated as follows:

$$softmax\,(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}, \quad \text{where} \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{10}$$

Classification networks predicting a binary label could make use of sigmoidal output units. Logistic *sigmoid* (sigma) and hyperbolic tangent *tanh* are examples of such units. The logistic *sigmoid* function gives a Bernoulli distribution of the layer output. To properly model this probability, the output is restricted in [0,1], and can be calculated as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \quad \text{where} \quad x \in \mathbb{R}. \tag{11}$$

The *tanh* activation is used to map input values into an output in [-1,1] and is obtained by the following formula:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \quad \text{where} \quad x \in \mathbb{R}. \tag{12}$$

Each of these nonlinearities has distinguishing properties that affect the training of the architecture, and the choice between them is case-dependent. Sigmoidal units saturate to a high value when their input x is very positive, and saturate to a low value when x is very negative, and are only strongly sensitive to their input when x is near zero. This can impede gradient-based learning. Another key difference to note is that *softmax* operates on an entire row or a vector of input, achieving what is known as the winner-takes-all effect. This means that the higher the probability one input of a vector gets, the lower the remaining values for the other elements will be, since the output has to sum up to one [42]. On the other hand, sigmoidal units are applied element-wise and map a single scalar independently into the output space.

1) NONLINEARITIES IN TRANSFORMER MODELS

In classic attention, the *softmax* function in 10 is used to map the Q, and the K scaled dot-product values from the vector space into a probability distribution. These probability distributions of the vector variable x are known as logits. In this scenario, they represent the learnable weights that allow the model to relate input tokens to one another to predict the next token. In addition to that, the softmax function plays a role in preventing the network from facing vanishing or exploding gradients during training.

Solutions to a number of problems have been presented in the realm of language processing and context dependency. In language modeling, [33] pointed out an undesirable effect they termed the bottleneck of softmax, where they hypothesize that natural language is high-rank, and that having a single function computing probabilities limits the expressiveness of language models. They illustrate that the embedding dimension d is usually set at a certain scale, e.g., 10^2, whereas the rank of a matrix A which represents possible logits can possibly be as high as 10^5, which is orders of magnitude larger than the embedding dimension d. With a *softmax* function the model learns a low-rank approximation of A. They argue that such approximation leads the network to lose the ability to model context dependency, both qualitatively and quantitatively. They address this by averaging several *softmax* distributions of the next token output probabilities. In [45], random positive orthogonal features were leveraged to approximate the *softmax* function. Their method has shown that this can effectively decrease the complexity in attention calculation. In addition to that, the imposed constraint of nonnegative outputs was a boost to the performance of their proposed method.

The choice of an activation function to be used as an output mapping function or as a hidden unit remains an active area of research. Although some units are a common choice for a certain type of network, there is still no definitive guideline for the optimal choice. Quite often, the choice of an activation unit is a trial-and-error process [42]. Although other activation units may be used, we focus on the activation functions mentioned above.

The mixture of softmax (MoS) technique, originally introduced to overcome the softmax bottleneck in LLMs [33], give inspiration to the idea of normalizing the spatial and edge encodings in graph-based models. Just as MoS addresses bottlenecks in representation space, spatial and edge encodings in graph architectures introduce additional structural information that must be effectively integrated without overwhelming the model. Applying *softmax* to normalize these encodings ensures they contribute meaningfully to the learned representations while preventing distortions caused by scale differences. In transformers, the positive values produced by the *softmax* function amplify the significance of the final attention scores, helping to clarify the relative importance of each input token. The advantage of using *softmax* lies in its ability to normalize input values based on their importance, ensuring that they sum to 1, regardless of the original range of the input vector x_i.

Section III explains our proposal, in which the properties of a given nonlinearity could be leveraged to further influence the role of spatial encoding in the Graphormer architecture.

III. METHODS

Our architecture builds on Graphormer [40], enhancing the integrated positional encodings and structural biases that better reflect the inductive priors of chemical graphs. The following subsections detail our technical contribution beyond the existing graph neural network variants. Fig. 1 illustrates an overview of the proposed architecture.

A. ARCHITECTURE OVERVIEW

We follow the general transformer paradigm with key adaptations for graphs based on the previously introduced Graphormer model. Defined by the following components:

- Node and edge embeddings are intialized from molecular graphs using atomic and bond features.

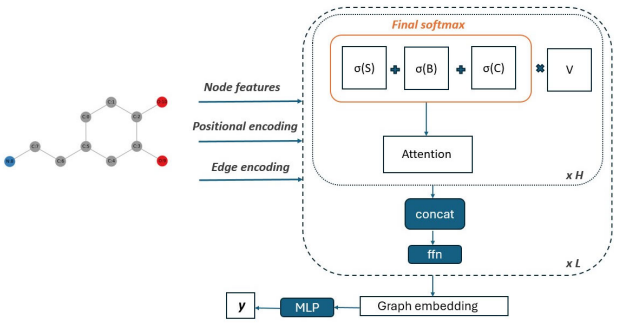


FIGURE 1. The Graphormer model with unified positional and edge biases.

- In Graphormer, self-attention is modified to to include PEs and edge descriptors introduced as bias. These biases provide information on the graph structure.
- The output of the final transformer layer is aggregated to generate a graph-level representation that could be used for graph classification or regression tasks.

B. REWRITING THE ATTENTION MECHANISM

We proceed to rewrite the formula 7 as a row-wise matrix equation instead of an element-wise matrix equation, since it is needed to define our attention mechanism. For a graph G with m nodes, let $A \in \mathbb{R}^{m\times m}$ be a matrix such that all elements are A_{ij} in 7. A can be described as follows:

$$A = S + B + C. \tag{13}$$

A row vector of A denoting the attention weights of a given node can be written as follows:

$$\boldsymbol{A_i} = \boldsymbol{S_i} + \boldsymbol{B_i} + \boldsymbol{C_i}, \tag{14}$$

where,

- $\boldsymbol{S_i} = [\frac{(h_i W_Q)(h_1 W_K)^\top}{\sqrt{d}}, \ldots, \frac{(h_i W_Q)(h_m W_K)^\top}{\sqrt{d}}]$,
- $\boldsymbol{B_i} = [b_{\phi(v_i,v_1)}, b_{\phi(v_i,v_2)}, \ldots, b_{\phi(v_i,v_m)}]$,
- $\boldsymbol{C_i} = [c_{i1}, c_{i2}, \ldots, c_{im}]$, where c_{ij} is described in 8.

With the formulations described previously, we can now introduce our model, which incorporates the concepts described in Section II-C and in 7.

C. UNIFYING THE DOMAINS OF THE POSITIONAL ENCODINGS

The values of the similarity matrix S, the positional or spatial encoding B, and the edge encoding C, can differ significantly in range due to representing distinct types of information and therefore having embeddings that are distinct in value from one another as explored in detail in Section IV. Additionally, since the PEs in the Graphormer are injected after similarity calculation rather than at the beginning, this makes the disparity in quantities more prominent. We argue that by simply moving the encodings, B and C, into the same numerical domain as the similarity score S, we could achieve an effect which emphasizes these properties in the overall attention score calculation. In order to give the three terms the same importance, we could consider normalizing them. This can be applied element-wise to each of the terms by utilizing a nonlinearity to bound those values to a single domain; thus 14 can be reformulated as follows:

$$A_{ij} = \sigma\left(S_{ij}\right) + \sigma\left(B_{ij}\right) + \sigma\left(C_{ij}\right), \tag{15}$$

where σ could describe a nonlinear function such as a *tanh* or a *sigmoid*.

Since these units are applied element-wise, they may disrupt the relationships between nodes. However, they effectively normalize the features and align them with the range of the similarity matrix S. To better preserve these relationships, it may be beneficial to use a row-wise operation that considers the relative contributions of all elements within a row. In this case, we propose using the *softmax* function to ensure that all three terms fall within the same range of values [0,1]:

$$\boldsymbol{A_i} = softmax\left(\boldsymbol{S_i}\right) + softmax\left(\boldsymbol{B_i}\right) + softmax\left(\boldsymbol{C_i}\right). \tag{16}$$

Furthermore, taking into account that using a *softmax* generates a distribution with correlated probabilities, a relation between each of the nodes is estimated based on their structural position. This not only magnifies the effect of these encodings, but does so by mapping their positional importance and assigning them meaningful values. The choice of *softmax* is intuitive when considering its original role in attention mechanisms. In LLMs, *softmax* assigns probability scores that determine the attention each token receives. A similar concept can be applied to graphs, where certain nodes or edge features hold greater significance than others. This approach allows the model to learn how individual nodes and edges contribute to the overall graph structure, which is particularly useful for tasks that depend on these individual contributions. Thus, we argue that *softmax* is a justified choice over other normalization methods. The processing of layers and attention heads remains similar to the original Graphormer.

IV. EXPERIMENTAL VALIDATION

A. THE DATASETS

To validate our proposal, experiments were carried out on a selection of public chemical datasets to predict a certain molecular property of a given molecule. Chemical compounds are represented by attributed graphs, where the graph nodes represent atoms and the edges represent bonds. In this way, a certain molecular property can be predicted by a graph regression task. The graphs of these chemicals were generated from SMILE strings. The node and edge attributes consisted of the encoded atom type, node degree, the shortest path distances, and edge features as presented in [40].

- **ESOL** [46], the set consists of 1128 molecules for the prediction of water solubility. The graphs of this dataset range between 2-55 nodes.
- **FreeSolv** [47], consists of both experimental and calculated hydration free energies of small neutral molecules

in water. The set used was comprised of 639 molecules represented as graphs ranging between 2-24 nodes.

- **Lipophilicity** [48], this set contains 4200 molecules. It measures the hydrophobicity, i.e., lipophilicity, which is the dissolvability of substance in nonpolar solvents. The graphs in this set range between 7-115 nodes.
- **PCQM4Mv2** This is a subset of the PubChem dataset [49] containing 3000 molecules for training, validation and testing. In this set, graphs contain 8-18 nodes, and each node is associated with a 9-dimensional feature (e.g., atomic number, chirality) and each edge comes with a 3-dimensional feature vector.

The molecular properties of the atoms and bonds were extracted using the RDKit cheminformatics open-source library.

B. HYPERPARAMETERS AND MODEL CONFIGURATION

Our model was trained end-to-end for graph regression tasks using the mean squared error (MSE) loss. Graphs and their properties were extracted from the Simplified Molecular Input Line Entry System (SMILES). Each dataset was divided into 80% for training and the remaining 20% was equally divided into validation and testing. The batch size was set to 32 for the ESOL and Lipophilicity datasets and 64 for FreeSolv. However, DeepSpeed memory optimization was used to automatically adjust batch size. All encodings and biases were jointly learned with the model parameters.The learning rate started at lr= 5e-5 with an AdamW optimizer, and Mean Squared Error (MSE) was used for evaluation. The model was evaluated with h= 2, 8 heads, l= 6 layers, and an embedding dimension d= 768. Training lasted 300 epochs: approximately 4 hours for ESOL, under 2 hours for FreeSolv, and nearly 2 days for Lipophilicity. The training was conducted on a single NVIDIA RTX A4000 GPU. The PyTorch Hugging Face[1] implementation of Graphormer was used under an Apache 2.0 license, with code and datasets available on GitHub.[2]

C. ANALYSIS OF THE DOMAINS OF THE ENCODING MATRICES

In this section, we empirically demonstrate that the matrices in 13 tend to operate in different numerical domains. As a result, the contribution of each of the three terms varies significantly. Consequently, omitting one or both terms, such as the spatial or edge encodings, may have minimal impact on the model's performance. Since these encodings carry substantially important information about the graph structure, the findings indicate that, in their current form, these encodings fail to take advantage of the rich information embedded within the graphs in an effective way.

[1]https://github.com/huggingface/transformers/tree/v4.29.0/src/transformers/models/graphormer

[2]https://github.com/ASCLEPIUS-URV/Graphormer-with-Softmax-Biases

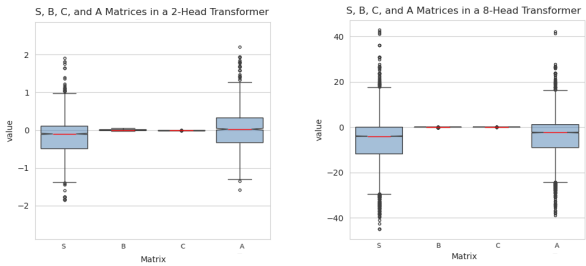


FIGURE 2. Box plots of the values obtained from one graph of the matrices contributing to the attention calculation, $A = S + B + C$. Left: The model with 2 heads, Right: The model with 8 heads.

Therefore, it is crucial to identify a suitable method to enable the network to learn the underlying structure of a given graph. This observation suggests the potential benefits of normalizing the values of these matrices and weighting their contributions to enhance their influence. Such adjustments could help the model better capture and utilize the relative features of the graph.

Fig. 2 shows the values of the similarity matrix, S, spatial encoding matrix B, and edge encoding matrix C obtained as explained by 7 from a single graph from the ESOL dataset. The left box plot was generated using a Graphormer with two heads, whereas the right box plot shows the same graph processed through an eight-head Graphormer.

In both variations of the architecture, we observe a significant difference in the standard deviation and mean between the matrix S and the matrices B and C. In particular, the minimum values of the matrices B and C tend to be higher than the mean of the S matrix, surpassing a substantial percentage of the values within S. It is also important to note that the matrices S and A occupy nearly identical value domains, with only a slight increase in A after the addition of the encodings.

Comparing both architecture variations (2 and 8 heads), we realize that the difference between S and the other two matrices becomes more pronounced in the 8-head architecture than in the 2-head architecture. In contrast, we consider similar behaviors of B and C in both architectures. We conclude that there is a tendency to increase the difference between the standard deviation and the mean in the matrix S and the matrices B and C when the number of heads increases. Thus, the problem of B and C having little to no contribution to the final value of the attention mechanism is more pronounced when the heads increase, which is the current tendency of recent architectures.

Similarly to Fig. 2, Fig. 3 shows the box plots of matrices $softmax(S)$, $softmax(B)softmax(C)$ obtained by our proposal with a 2-head architecture. As expected, the domains of the values of the three first matrices are within 0 and 1. In this case, the mean values of the three matrices are closer, although matrix $softmax(B)$ shows a larger standard deviation. In this model, the values of A are clearly larger than the values of the matrices $softmax(S)$, $softmax(B)$ and

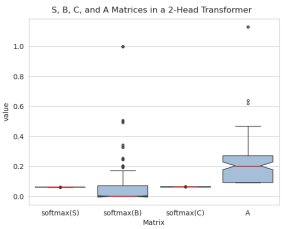


FIGURE 3. Box plots of the values obtained from one graph showing the matrices contributing to the attention calculation, $A = softmax(S) + softmax(B) + softmax(C)$.

TABLE 1. MSE generated by the original model (first column), without edge encoding (second column), the original model without the spatial encoding (third column), and the original model and the original model without any encoding (last column).

Dataset/ Model	$A = S + B + C$ (Eq. 13)	$A = S + B$	$A = S + C$	$A = S$
Esol	1.1312	1.1468	**1.0314**	1.1468
Freesolv	3.5325	**3.5320**	3.5368	3.5368
Lipophilicity	**1.3949**	**1.3949**	1.3951	**1.3949**
PCQM4Mv2	0.3781	0.4117	**0.2327**	0.4117

softmax(C) since they contribute almost similarly to the final result. As mentioned previously, we have visualized these box plots given several architectures with different numbers of heads, and in this case, we observed that this effect remains consistent regardless of the number of heads used. Thus, the difference between matrix domains detected in the original architecture will not be observed with the increase in number of heads.

D. ANALYSIS OF THE IMPORTANCE OF THE ENCODINGS

Table 1 presents the MSE generated by the original model (first column), the original model without edge encoding (second column), the original model without spatial encoding (third column), and the original model without any encoding (last column). The results show that the benefit of the encoding is not consistent across all datasets. There were cases where using spatial encoding yielded the same error score as not using any encoding. This effect can be observed with the Lipophilicty dataset. This suggests that the impact of the spatial encodings is minimal, which further supports our belief that the encoding value might get lost when added to the attention score. In most cases, we observed that using both spatial B and edge encodings C slightly decreased the error, indicating a potential benefit of combining these terms. In the case of the Freesolv data, the use of the spatial encoding by itself reduced the error marginally of $\sim 0.01\%$. This indicates that these encodings have a negligible effect on this dataset, whereas they could be considered noise in other cases. The contrary was seen with the PCQM4Mv2 data where including edge encodings led to a significant improvement in the model's performance. This variability suggests that the impact of encodings may depend on the dataset's specific characteristics.

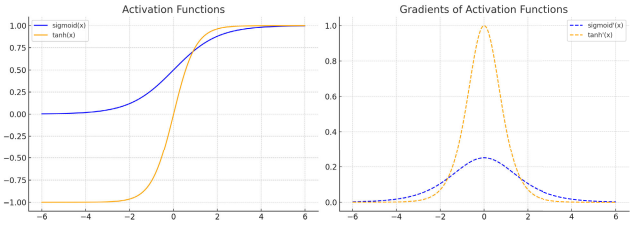


FIGURE 4. Gradients of *tanh* and *sigmoid* functions.

TABLE 2. Test MSE obtained by using the original model [40] (first column), our proposal with *softmax* (second column), our proposal with tangent hyperbolic (third column), and our proposal with *sigmoid* (last column) on four datasets.

Dataset/Model	Original Model	*softmax*	*tanh*	*sigmoid*
ESOL	0.7143	**0.6150**	0.9513	0.9420
Freesolv	2.8399	**2.0601**	4.4538	4.3138
Lipophilicity	0.8133	**0.7620**	1.4243	1.4100
PCQM4Mv2	0.4073	**0.1694**	0.3094	0.2819

E. ANALYSIS OF INCORPORATING NONLINEARITIES IN THE ENCODINGS

Table 2 shows the MSE given four models obtained from four datasets. The original model [40] in the first column and three combinations of our model in the other three columns, depending on the applied nonlinear function: *softmax* (second column), tangent hyperbolic, *tanh*, (third column) and *sigmoid* (last column). From the computed MSE, we conclude that the effects of normalizing the biases are prominently visible in certain cases more than others. For instance, in the ESOL and PCQM4Mv2 datasets, a clear reduction in error is observed when all fields are normalized to the same numerical domain in comparison with error scores presented in Table 1. The comparison between the original model and the one modified with *softmax* shows a promising improvement. Consistent improvements were observed across all datasets, with a particularly notable $\sim 27.45\%$ decrease in MSE on the Freesolv dataset. However, not all nonlinearities yielded optimal results. Applying functions such as *sigmoid* and *tanh* to the similarity scores and encodings did not consistently improve the performance of the model in most datasets, except for the PCQM4Mv2 dataset. Although all three functions that have been incorporated introduce nonlinearity that helps the model learns complex patterns, the behavior of these functions differ from one function to another. In the case of *tanh*, we could explain this outcome by the fact that its gradients in generally larger than that of the *sigmoid* function, especially around 0. As shown in Fig. 4, the gradient, *tanh'*, peaks at 1 when $x = 0$, whereas *sigmoid'* peaks at 0.25. This could lead to larger weight updates when using *tanh*, which could affect the model's ability to learn and converge. Additionally, the values of B and C are small in magnitude (with values nearing 0). This means that applying *tanh* or *sigmoid* could further squash the differences between these values, which might weaken the ability of the model to learn

TABLE 3. Test MSE obtained from different graph learning baseline models, compared with our softmax Graphormer.

Dataset/Model	GCN	GAT	GT	Ours
ESOL	1.8957	2.6480	1.4551	**0.6150**
Freesolv	5.1528	13.7937	5.9462	**2.0601**
Lipophilicity	1.2397	1.2822	0.9297	**0.7620**
PCQM4Mv2	0.5098	0.4050	0.3895	**0.1694**

nuanced distinctions. This effect is not observed with *softmax* since its applied vector-wise, preserving relative differences.

Fig. 5 shows eight scatter plots obtained by the four datasets. The vertical axes are the predicted global values and the horizontal axes are the ground truth values. The results compare the predictions of the original model and our model using *softmax*. In the case of the FreeSolv data, the predicted values given by the original model spanned a different range of values $[-17.5, -2.5]$, Fig. 5 (c), compared to the range of the ground truth values $[-20, 0]$, Fig. 5 (d), to which the modified model adhered. Despite the presence of outliers in the predictions, we could argue that our model tends to generate a narrower and more diagonal cloud of points. Although the difference between these scatter plots may not be clear at first glance, the MSE values in Table 2 support the efficacy of our modification. These outcomes consolidate the premise that the unification of widely varying domains can positively impact the final attention score. Additionally, the variance in results across the different test sets confirms the sensitivity of transformer models to changes in graph structure and hyperparameters. It is worth mentioning that standard normalization techniques, e.g., were tested in the initial stages of these experiments. However, they did not yield competitive results and were not included for the sake of brevity. This enhancement of performance using softmax could be explained by its use of the exponential function. This amplifies larger values relative to smaller ones, emphasizing the most significant values. This helps in scenarios like attention mechanisms, where it is important to highlight the most relevant inputs while suppressing others. Additionally, *softmax* ensures all outputs are non-negative, which aligns with probability distribution property that the attention model relies on. Normalization techniques like Z-score can result in negative values, which may not be meaningful in this context.

F. COMPARISON WITH PRIOR ARCHITECTURES

Table 3 presents the MSE obtained from baseline graph learning architectures. In the absence of known optimal hyperparameters, the same configuration of Graphormer was adopted. Comparison experiments can be found in the same Github repository.[3] We observe that the proposed model consistently outperforms GCN, GAT, and GT across the datasets tested. The relatively poor performance of GCN can be attributed to its original design for node classification on a single, large graph. GCNs aggregate information from local neighborhoods and lack support for edge attributes, which limits their ability to model fine-grained interactions in molecular graphs. Moreover, GCNs tend to suffer from performance degradation when deepened, due to oversmoothing and vanishing gradients.

GAT models incorporate attention mechanisms, allowing nodes to weigh the importance of their neighbors. However, this attention is computed only over local neighborhoods, which restricts the model's ability to capture dependencies of a longer range and global context. These aspects are crucial when learning molecular graphs. Additionally, standard GATs do not encode relative or absolute positional information, such as node order or spatial structure, further limiting their expressiveness in tasks where geometry matters.

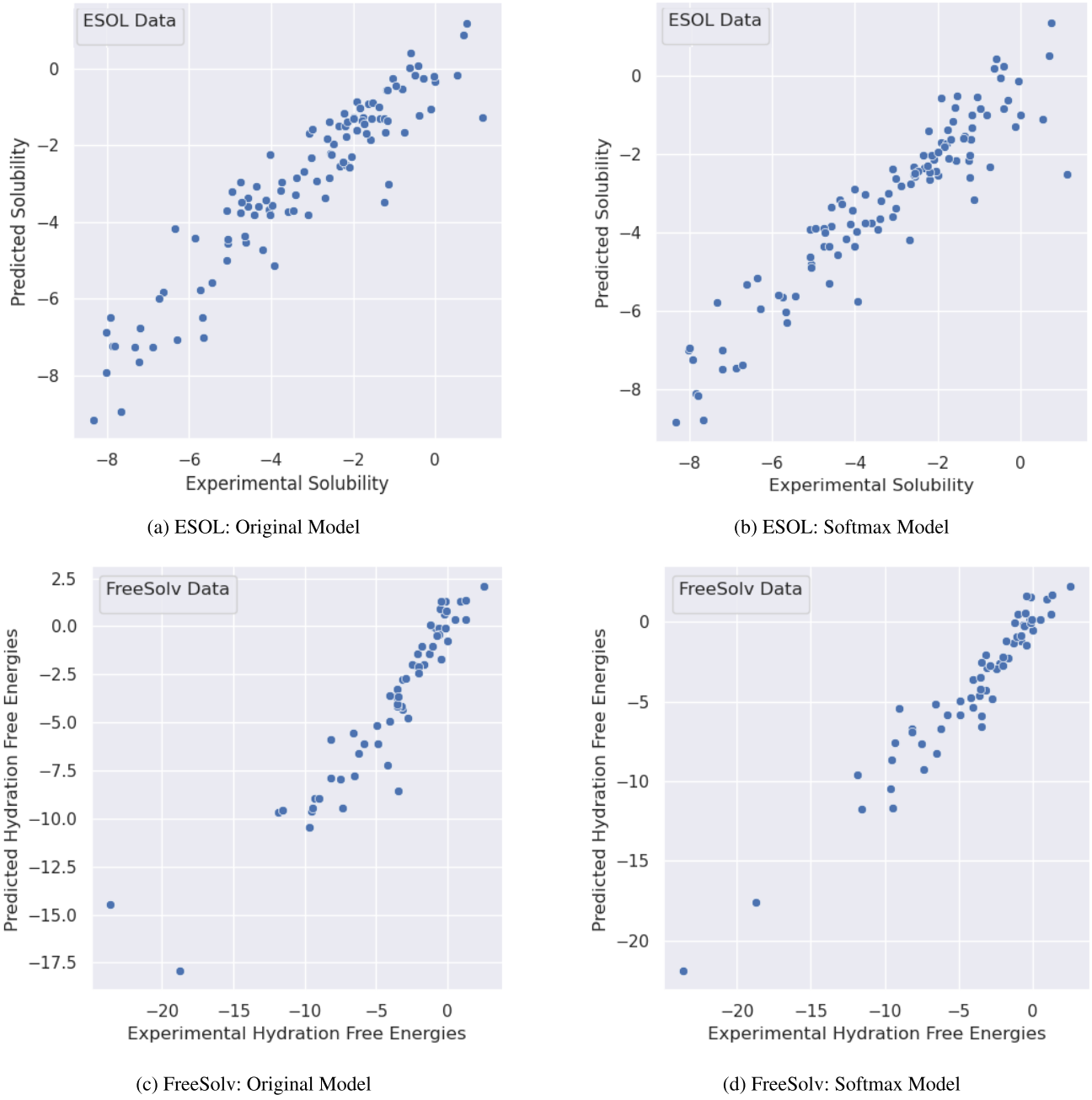
GT models on the other hand, use full attention and incorporate positional encodings derived from graph Laplacian eigenvectors. While this allows for global context aggregation, the absence of domain-specific inductive biases, i.e., spatial distance, edge encoding, or centrality, can limit performance in molecular property prediction. Our architecture augments the existing tailored structural encodings, offering a stronger bias toward relevant molecular characteristics.

Finally, although these baseline models can achieve competitive performance with extensive hyperparameter tuning or pretraining, this process is computationally expensive, particularly on smaller datasets that are difficult to learn without pretraining [50]. In contrast, our methods reduce reliance on such tuning while effectively enhancing the learning of of both structural and global semantics, mitigating issues like oversmoothing and oversquashing.

G. COMPLEXITY ANALYSIS

The core of our model is the introduction of a nonlinear function in each of the three terms of the attention mechanism as means of normalization and highlighting feature importance. The *softmax* function emphasizes relative differences between elements, which might not always be desirable. For example, if all values in a row are similar, softmax will still artificially amplify small differences, potentially introducing noise. Furthermore, transformer models incur longer training time that scales quadratically with the embedding dimension and input size [30]. Since the *softmax* equation is computed per row of each one of the three matrices S, B and C, the computational cost of *softmax* is linear with respect to the number of elements of the vector x_i, as can be deduced in Eq. 10. Thus, the training of our model increases the original attention computational cost of $O(m^2 \cdot d)$ by an additional $O(m)$. Although this overhead is modest, it may pose a concern for large graphs or real-time applications. Nevertheless, this increase in runtime could be offset by a reduction in the number of training epochs, as the proper incorporation of information into the attention-based architecture may enable the model to reach a stable minimum more quickly.

[3] https://github.com/ASCLEPIUS-URV/Graphormer-with-Softmax-Biases



(a) ESOL: Original Model

(b) ESOL: Softmax Model

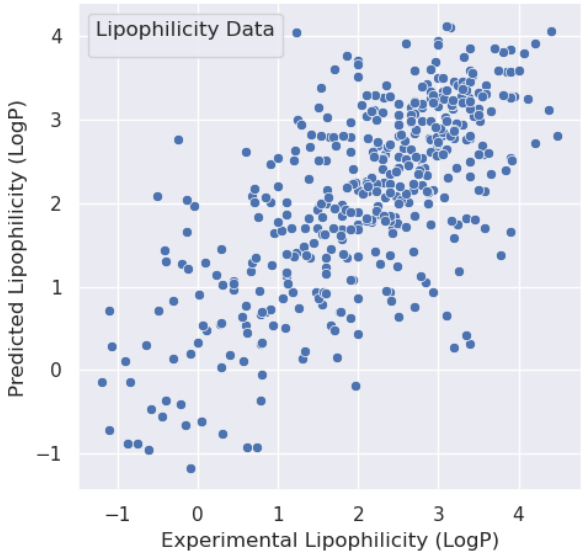
(c) FreeSolv: Original Model

(d) FreeSolv: Softmax Model

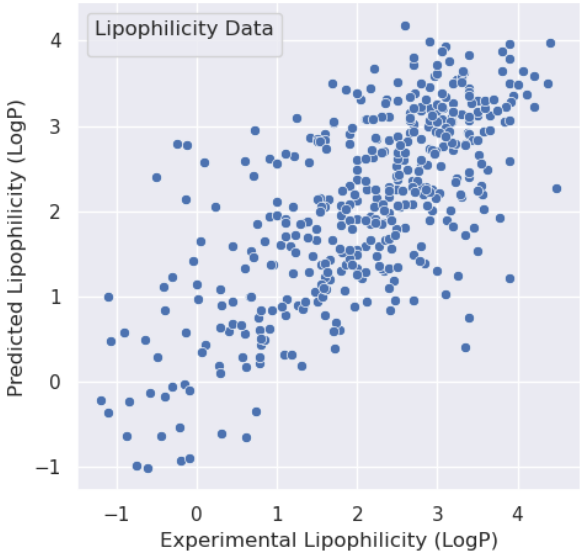
FIGURE 5. Scatter plots showing the real vs. predicted values of the original model (left) and our proposal (right) across multiple datasets.

V. LIMITATIONS

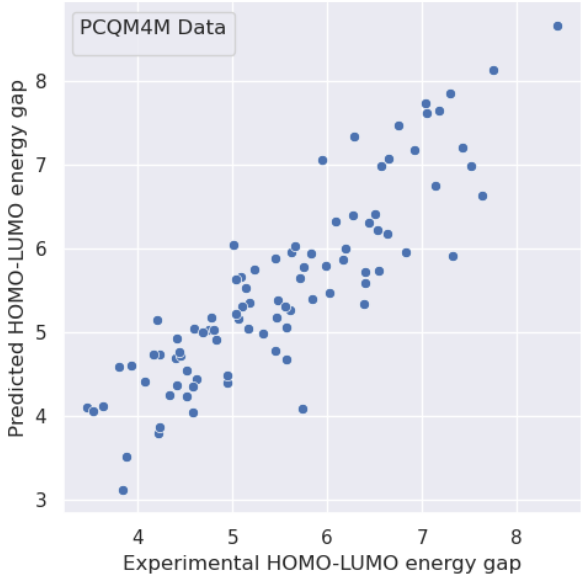
From a theoretical perspective, we assume that the three matrices, S, B, and C, provide valuable information to the attention mechanism. However, in certain applications, depending on the nature of the graphs, incorporating one of these matrices may not yield significant benefits. This was observed most prominently with the Freesolv and Lipophilicity datasets in Table 1. Graph data varies in structure, and the relationships between its components differ between datasets. As a result, no single combination of encodings guarantees optimal performance in all cases. While unifying the domains of the attention mechanisms and encodings improves overall model performance, it remains essential to customize these embeddings for each specific scenario. In addition to that, during the training process we have observed that the Graphormer model is quite sensitive to configuration and hyperparameter settings. The performance of the model on a single dataset varies greatly depending on the combination of number of heads, layers, embedding dimension and training parameters and the dataset



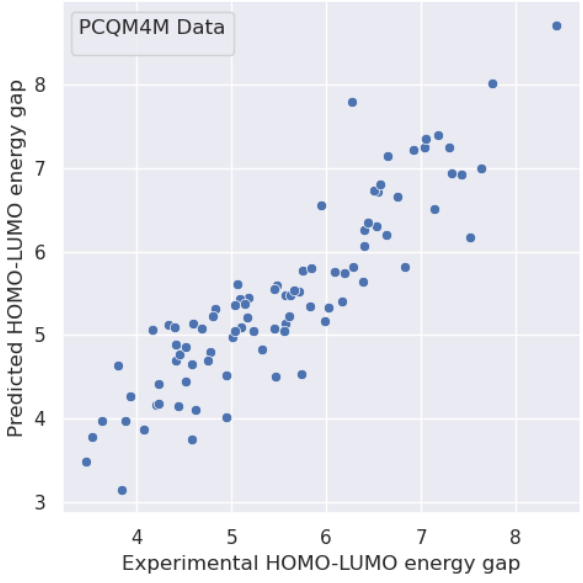
(e) Lipophilicity: Original Model



(f) Lipophilicity: Softmax Model



(g) PCQM: Original Model



(h) PCQM: Softmax Model

FIGURE 5. ***(Continued.)*** **Scatter plots showing the real vs. predicted values of the original model (left) and our proposal (right) across multiple datasets.**

size. This means that the hyperparameter space is large and exhaustive search is needed to find the optimal set of configurations to train a model on a chosen dataset. This observation can be extended to other graph learning methods when examining the varying results between one model and another with the same dataset. Finally, while our model outperformed others across all datasets, it is important to note that transformer-based models generally struggle to learn effectively from small datasets [51], which characterizes those used in this study.

VI. CONCLUSION

When compared to other graph learning baseline methods, better results are achieved with an out-of-the-box Graphormer since it is tailored to molecular data. This is thanks to its incorporation of spacial distance and edges between nodes known as positional encodings. We analyze the role of positional encodings used in the graph transformer architecture model Graphormer. Ablation studies indicate that the effectiveness of both node and edge encodings varies across different datasets. Although some encoding

combinations may slightly improve performance, these effects are generally small and inconsistent. Therefore, the inclusion of encodings should be tailored to the specific dataset.

We also propose modifying the Graphormer model, emphasizing the importance of spatial and edge encodings in calculating the overall attention score. Our findings show that unifying the range of these encodings within the self-attention mechanism, alongside the initial similarity score, enhances the final attention output. By ensuring that the critical properties of the graph are properly weighted, the model assigns greater significance to the relevant features, improving its expressive power. This unified approach results in more effective embeddings, optimized for key graph-related tasks. In future work, additional datasets could be included to address a broader range of applications. A wider variety of graph structures, including graphs with different orders, varying numbers of node attributes, or differing quantities of graphs, would provide a more comprehensive insight. Finally, although our proposed method showed noticeable improvement in graph prediction tasks, other normalization methods such as layer or batch normalization could be tested.

Future research will focus on further analyzing the impact of these encodings on the attention mechanism while maintaining low computational costs. This can be achieved by introducing learnable weights for each encoding term, enabling the model to determine their relative contribution based on the specific task. This approach would allow for the dynamic selection of the most suitable encoding type to enhance graph embedding learning. Another promising direction is to apply these modified embeddings to generative tasks, broadening the model's applications beyond predictive tasks. Continuous refinement of these methods will expand the utility of the model and improve performance across a wider range of applications.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] X.-M. Zhang, L. Liang, L. Liu, and M.-J. Tang, "Graph neural networks and their current applications in bioinformatics," *Frontiers Genet.*, vol. 12, Jul. 2021, Art. no. 690049.

[2] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *J. Chem. Inf. Comput. Sci.*, vol. 28, no. 1, pp. 31–36, Feb. 1988.

[3] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

[4] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–11.

[5] A. Apicella, F. Isgrò, A. Pollastro, and R. Prevete, "Adaptive filters in graph convolutional neural networks," *Pattern Recognit.*, vol. 144, Dec. 2023, Art. no. 109867.

[6] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, May 2020.

[7] J. Wang, Y. Guo, L. Yang, and Y. Wang, "Heterophily-aware graph attention network," *Pattern Recognit.*, vol. 156, Dec. 2024, Art. no. 110738.

[8] Y. Sun, Y. Duan, H. Ma, Y. Li, and J. Wang, "High-frequency and low-frequency dual-channel graph attention network," *Pattern Recognit.*, vol. 156, Dec. 2024, Art. no. 110795.

[9] M. Balcılar, P. Héroux, B. Gaüzère, P. Vasseur, S. Adam, and P. Honeiné, "Breaking the limits of message passing graph neural networks," in *Proc. 38th Int. Conf. Mach. Learn.*, Jul. 2021, pp. 599–608.

[10] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023, *arXiv:2303.10993*.

[11] W. L. Hamilton, "Graph representation learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 14, no. 3, pp. 1–159, 2020.

[12] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9266–9275.

[13] L. Rampek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," 2023, *arXiv:2205.12454*.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[15] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang, "GIT: A generative image-to-text transformer for vision and language," 2022, *arXiv:2205.14100*.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–11.

[17] X. Liu, H. Lu, J. Yuan, and X. Li, "CAT: Causal audio transformer for audio classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[18] P. Xu, X. Zhu, and D. A. Clifton, "Multimodal learning with transformers: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12113–12132, Oct. 2023.

[19] Y. Duan, N. Chen, P. Zhang, N. Kumar, L. Chang, and W. Wen, "MS2GAH: Multi-label semantic supervised graph attention hashing for robust cross-modal retrieval," *Pattern Recognit.*, vol. 128, Aug. 2022, Art. no. 108676.

[20] J. Tang, J. Hu, W. Huang, S. Shen, J. Pan, D. Wang, and Y. Ding, "Spatio-temporal graph convolution transformer for video question answering," *IEEE Access*, vol. 12, pp. 131664–131680, 2024.

[21] X.-B. Ye, Q. Guan, W. Luo, L. Fang, Z.-R. Lai, and J. Wang, "Molecular substructure graph attention network for molecular property identification in drug discovery," *Pattern Recognit.*, vol. 128, Aug. 2022, Art. no. 108659.

[22] Y. Raisinghani, A. Shah, and M. Rahevar, "Stacked graph transformer for HIV molecular prediction," in *Proc. 2nd Int. Conf. Commun., Comput. Ind. 4.0 (C2I4)*, Dec. 2021, pp. 1–6.

[23] V. Mazzia, S. Angarano, F. Salvetti, F. Angelini, and M. Chiaberge, "Action transformer: A self-attention model for short-time pose-based human action recognition," *Pattern Recognit.*, vol. 124, Apr. 2022, Art. no. 108487.

[24] Y. Li, D. Chen, T. Tang, and X. Shen, "HTR-VT: Handwritten text recognition with vision transformer," *Pattern Recognit.*, vol. 158, Feb. 2025, Art. no. 110967.

[25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–17.

[26] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, *Graph Transformer Networks*. Red Hook, NY, USA: Curran Associates, 2019.

[27] L. Mller, M. Galkin, C. Morris, and L. Rampek, "Attending to graph transformers," 2024, *arXiv:2302.04181*.

[28] M. A. K. Raiaan, M. S. H. Mukta, K. Fatema, N. M. Fahad, S. Sakib, M. M. J. Mim, J. Ahmad, M. E. Ali, and S. Azam, "A review on large language models: Architectures, applications, taxonomies, open issues and challenges," *IEEE Access*, vol. 12, pp. 26839–26874, 2024.

[29] M. Jiang, G. Liu, Y. Su, and X. Wu, "Self-attention empowered graph convolutional network for structure learning and node embedding," *Pattern Recognit.*, vol. 153, Sep. 2024, Art. no. 110537.

[30] A. Shehzad, F. Xia, S. Abid, C. Peng, S. Yu, D. Zhang, and K. Verspoor, "Graph transformers: A survey," 2024, *arXiv:2407.09777*.

[31] Y. Shi, A. Zhang, E. Zhang, Z. Liu, and X. Wang, "ReLM: Leveraging language models for enhanced chemical reaction prediction," in *Proc. Findings Assoc. Comput. Linguistics, EMNLP*, Singapore, Dec. 2023, pp. 5506–5520.
[32] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 21618–21629.
[33] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, "Breaking the softmax bottleneck: A high-rank RNN language model," 2017, *arXiv:1711.03953*.
[34] X. Zhou, Z. Ren, S. Zhou, Z. Jiang, T. Yu, and H. Luo, "Rethinking position embedding methods in the transformer architecture," *Neural Process. Lett.*, vol. 56, no. 2, p. 41, Feb. 2024.
[35] V. Prakash Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," 2020, *arXiv:2012.09699*.
[36] J. Yeom, T. Kim, R. Chang, and K. Song, "Structural and positional ensembled encoding for graph transformer," *Pattern Recognit. Lett.*, vol. 183, pp. 104–110, Jul. 2024.
[37] H. Wu, A. Yip, J. Long, J. Zhang, and M. K. Ng, "Simplicial complex neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 1, pp. 561–575, Jan. 2024.
[38] Y. Zhang, Q. Wang, D.-W. Gong, and X.-F. Song, "Nonnegative Laplacian embedding guided subspace learning for unsupervised feature selection," *Pattern Recognit.*, vol. 93, pp. 337–352, Sep. 2019.
[39] V. P. Dwivedi, A. T. Luu, L. Thomas, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–7.
[40] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T. Liu, "Do transformers really perform bad for graph representation?" in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 1–8.
[41] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, Jun. 1962, doi: 10.1145/367766.368168.
[42] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[43] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 901–909.
[44] Z. Chen and P.-H. Ho, "A generic shift-norm-activation approach for deep learning," *Pattern Recognit.*, vol. 109, Jan. 2021, Art. no. 107609.
[45] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," 2020, *arXiv:2009.14794*.
[46] J. S. Delaney, "Esol: Estimating aqueous solubility directly from molecular structure," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 3, pp. 1000–1005, 2004.
[47] D. L. Mobley and J. P. Guthrie, "FreeSolv: A database of experimental and calculated hydration free energies, with input files," *J. Comput.-Aided Mol. Design*, vol. 28, no. 7, pp. 711–720, Jul. 2014.
[48] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "MoleculeNet: A benchmark for molecular machine learning," 2017, *arXiv:1703.00564*.
[49] M. Nakata, "The pubchemqc project: A large chemical database from the first principle calculations," *AIP Conf. Proc.*, vol. 1702, Dec. 2015, Art. no. 090058.
[50] R. Zhu, Z. Tao, Y. Li, and S. Li, "Automated graph learning via population based self-tuning GCN," 2021, *arXiv:2107.04713*.
[51] R. Shao and X.-J. Bi, "Transformers meet small datasets," *IEEE Access*, vol. 10, pp. 118454–118464, 2022.

SARAH A. FADLALLAH received the B.Sc. degree in information systems from the Bayan College of Science and Technology, Khartoum, Sudan, in 2012, and the M.Sc. degree in artificial intelligence and information security from Universitat Rovira i Virgili, Tarragona, in 2021, where she is currently pursuing the Ph.D. degree.

She joined Universitat Rovira i Virgili, where she is researching graph-based models for molecular generation and property prediction. From 2021 to 2022, she completed an Internship as a Junior Data Scientist with Zummit Infolabs. She has also completed a secondment with the Institute of Science Tokyo, Tokyo, Japan. She is the author of a journal article and three conference papers. Her research interests include machine learning, graph neural networks, computer vision, and ethical AI.

FRANCESC SERRATOSA was born in Barcelona, Catalonia, Spain, in 1967. He received the degree in computer science engineering and the Ph.D. degree from Universitat Politècnica de Catalunya, in 1993 and 2000, respectively.

He is currently a full-time Professor of computer science with Universitat Rovira i Virgili, Tarragona, Catalonia. He is the Principal Researcher of the Catalan Research Group 2021SGR-00111: "Asclepius: Smart Technology for Smart Healthcare". He is leading two European projects at Universitat Rovira i Virgili: SbD4Nano (H2020-NMBP-TO-IND-2019-862195) and NanoInformaTIX (H2020-NMBP-14-2018-814426). Moreover, he is also leading the Spanish Project NextPandemics (PID2022-138327OB-I00). He has advised 13 Ph.D. theses. He has published almost 200 articles. He is an active reviewer in research projects, congresses, and journals. He appears in the list of the 2% most influential researchers in the world, presented in "Updated science-wide author databases of standardized citation indicators, Plos Biology," in 2020, and "Updated science-wide author databases of standardized citation indicators," Elsevier BV, in 2021. He is an active researcher in computer vision, machine learning, and artificial intelligence applied to biotechnology and aquaculture.

CARME JULIÀ received the B.Sc. degree in mathematics from Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 2002, and the Ph.D. degree in computer science from Universitat Autònoma de Barcelona (UAB), Cerdanyola del Vallés, Spain, in 2008.

She arrived with the Computer Engineering and Mathematics Department (DEIM), Universitat Rovira i Virgili (URV), Tarragona, Spain, in 2008. She has been an Associate Professor, since 2019. Her lines of research are aimed at applying machine learning and deep learning to the field of health, either with images or with graphs.