DualGraphormer: towards better molecular properties prediction by leveraging edge features for node-edge fusion and switch

Xiao Xiao
Department of Biostatistics
Yale University
New Haven, CT 06510
xiao.xiao.xx244@yale.edu

Abstract

In the evolving landscape of graph deep learning, the prediction of molecular properties remains a pivotal challenge. Though many graph transformers have demonstrated a relatively good performance on molecular property prediction, they still suffer from inadequate encoding of edge features and lack of an efficient way to update node features by incorporating edge features. And on small dataset, graph transformer without pretraining on large dataset would have a very poor performance. In this paper, we introduce DualGraphormer, a novel architecture that significantly enhances molecular properties prediction by innovatively leveraging edge features for node-edge fusion and switch mechanisms. Our approach uniquely combines the strengths of transformer models with graph neural networks, simultaneously update node and edge features, and integrating them together to better represent the graph features. We evaluate our model on the ogb-molhiv dataset, and prove that our model has outperformed graphormer when not pretrained on large-scale datasets.

1 Introduction

The accurate prediction of molecular properties is a cornerstone of computational chemistry, material science, and drug discovery. Traditional methods often struggle with the intricate complexity of molecular interactions, particularly in how they integrate and interpret node (atom) and edge (bond) features within molecular graphs. This limitation poses a significant bottleneck in advancing applications like drug design, where understanding subtle atom interactions is crucial.

Current graph deep learning models, including message passing GNN[1] and graph transformers[2]. despite their advancements, predominantly treat node and edge features independently, failing to capture the holistic nature of molecular interactions. This discrete treatment results in a loss of critical information, hampering the ability to accurately predict complex molecular properties. The challenge, therefore, lies in devising a model that can seamlessly integrate these features, providing a more comprehensive understanding of molecular structures.

The ability to accurately predict molecular properties has profound implications. In pharmaceuticals, it can expedite drug discovery and enhance drug efficacy. In material science, it leads to the development of innovative materials with desired properties. Therefore, improving the predictive accuracy of molecular properties is not just a theoretical challenge but a practical necessity with far-reaching consequences.

We introduce DualGraphormer, a novel graph deep learning architecture designed to overcome the limitations of existing models. It features a unique Node-Edge Fusion module that integrates node and

Preprint. Under review.

edge features more effectively than ever before. The model also incorporates a Switch mechanism, which innovatively treat original edges as nodes in the switched graph and original nodes as edges. We apply our switching method in transformer models, known for their effectiveness in handling sequential data.

We evaluate our model on the ogb-molhiv dataset[3], and prove that our model has outperformed graphormer when not pretrained on large-scale datasets. This demonstrates that adding the switch mechanism to graph transformer would improve the performance of graphormer in terms of predicting and make the graph transformer model to converge faster in small datasets.

2 Related works

2.1 Graph transformers

Recent years have seen the success of the Transformer architecture in natural language processing[4] and computer vision[5], and it is natural to consider that Transformers will also perform well on graph deep learning tasks. Unlike traditional convolutional-based graph neural networks that can only receive information from limited reception field, the transformer can extract long-range relationships by self-attention. Also, the transformer can avoid the occurrence of over-smoothing in traditional graph neural networks. And with the help of the class token, we can still gather information from different nodes without using the graph pooling operation that may let the neural network lose certain information. Currently, Graph Transformers can be largely divided into two categories, with and without message passing. For the message-passing graph transformers, like SAN[6] and SignNet[7], they assume that neighbor nodes in the graphs are relatively similar to each other in terms of node features, but this is not often the case. And for the Graph Transformers without message passing, like Graphormer with shortest path embedding, they tend to perform badly on small datasets and require a large dataset to converge. Furthermore, current Graph Transformers focus on the updating of node features, but the contribution of edge features to the graph-level prediction task and the updating of edge features is ignored.

2.2 Graph neural network with message passing

A critical advancement in GNNs is the incorporation of message-passing mechanisms, which enable nodes in a graph to 'communicate' and exchange information, thereby refining their representations. The concept of message passing in GNNs was popularized by Graph Convolutional Network (GCN) proposed by Kipf and Welling (2016)[8], which simplified the convolution process on graphs and improved scalability and efficiency. Another notable advancement is the Graph Attention Network (GAT) introduced by Veličković et al. (2017)[9], which integrated attention mechanisms into the message-passing process, allowing for more flexible and dynamic feature aggregation from neighboring nodes.

More recent works have sought to enhance the message-passing process by incorporating edge features more effectively. For instance, the work by Schlichtkrull et al.[10] on modeling relational data with Graph Convolutional Networks introduced a way to include different types of relations (edges) in the message-passing process. This approach is particularly relevant in molecular graphs, where bond types significantly influence molecular properties.

However, despite these advancements, a gap remains in the seamless integration of node and edge features within the message-passing framework. Many existing models still treat these features somewhat independently, leading to insufficient representations of graph-structured data, especially in complex molecular structures. Our work with DualGraphormer addresses this gap by introducing a novel node-edge fusion mechanism within the message passing framework, significantly enhancing the accuracy of molecular property predictions.

2.3 Utilizing multi-dimensional edge features

In graph neural networks (GNNs), the effective utilization of edge features, especially multi-dimensional ones, is crucial for accurately modeling complex systems like molecular structures. In molecular modeling, the importance of multi-dimensional edge features is particularly pronounced due to the complex nature of chemical bonds. Previous works that used multi-dimensional edge fea-

tures usually focus on how to use edges to optimize message passing. These works include r-GAT[11] and r-GCN[10], which build multiple attention types for multiple dimensions of edge features; and EGNN[12], which adaptively encodes multiple edge features in different layers. However, these methods only use edge features to update edge features, regardless of the influence of node features to its connected edge features. However, CensNet[13] can solve this problem well by treating edges in the original graph as nodes in the graph after node-edge-switch.

3 Method

3.1 Node edge switch

We assume that we have an undirected graph G input with multi-dimensional edge E features and multi-dimensional node V generated by our connection analyzer. We define the input elements of our CensNet block as below.

- Let $\mathbf{X} \in \mathbb{R}^{N_V \times D_V}$ be the node feature matrix, where N_V refers to the number of nodes in the graph, and D_V refers to the number of dimensions of node features.
- Let $\mathbf{Z} \in \mathbb{R}^{N_E \times D_E}$ be the edge feature matrix, where N_E denotes the number of edges, and D_E refers to the number of dimensions of the edge features.
- Let $\mathbf{A}_V \in \mathbb{R}^{N_V \times N_V}$ be the node adjacency matrix, and the values in this matrix are continuous instead of only 0 or 1.
- Let $\mathbf{A}_E \in \mathbb{R}^{N_E \times N_E}$ be the edge adjacency matrix. The diagonal elements of the matrix are always non-zero, and the values in this matrix are discrete.
- Let $\mathbf{H}_v^{(l)}$ be the l-th hidden layer of the vertex features, and let $\mathbf{H}_E^{(l)}$ be the l-th hidden layer of the edge features. Specifically, $\mathbf{H}_v^{(l)}$ equals to $\mathbf{X}$ and $\mathbf{H}_E^{(l)}$ equals to $\mathbf{Z}$ at the 0-th layer.

The edge adjacency matrix $\mathbf{A}_E \in \mathbb{R}^{N_E \times N_E}$. There is a connection between $edge_i$ and $edge_j$ if $start_i = start_j$ or $start_i = end_j$ or $end_i = start_j$ or $end_i = end_j$, where $start_i$ and end_i refer to the starting vertex and ending vertex of $edge_i$, respectively. If there is a connection between $edge_i$ and $edge_j$, then $A_{E,ij} = 1$, otherwise $A_{E,ij} = 0$. In practice, we integrate A_E and Z to generate the edge features for original graph $E_V \in R^{N_V \times N_V \times D_E}$, and respectively, we have the edge features for the switched graph $E_E \in R^{N_E \times N_E \times D_V}$.

3.2 Graph attention network with multiple edge features

We present an enhanced Graph Attention Network (GAT) architecture designed to process graph-structured data with multiple edge features. This implementation leverages the powerful representational capabilities of attention mechanisms in handling complex relational data.

3.2.1 Normalization of the Adjacency Matrix

The normalization of the adjacency matrix is a crucial step in our approach. It is performed using the following formula:

$$A' = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$$

where A is the original adjacency matrix, I is the identity matrix, and D is the diagonal node degree matrix. This normalization step enhances the stability and efficiency of our model.

3.2.2 Attention Mechanism

The core of our GAT layer is the attention mechanism, which computes the importance of each node to a central node in the graph. The attention coefficients are computed using a shared attentional mechanism.

$$e_{ij} = \text{LeakyReLU}(W \cdot \text{concat}(h_i, h_j, e_{ij}))$$

where h_i and h_j are the feature vectors of the nodes, e_{ij} is the edge feature, and W is the weight matrix of the linear transformation applied to each node. The attention coefficients are normalized across all choices of j using the softmax function:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

And similar to previous works, we apply multi-head attention to our model.

3.3 Node edge fusion

In this section, we update the node features by a linear transformation of itself and its connected edges:

$$x_i = \text{Linear1}(x_i || \text{Linear2}(\sum_{j \in N(x_i)} e_{ij}))$$

where $||$ refers to tensor concatenation, $N(x_i)$ refers to the neighbors of node x_i. Linear1 is a linear transformation from D_V to D_V, and Linear2 is another linear transformation from D_E to D_V.

3.4 Graphormer

3.4.1 Centrality Encoding

This encodes node centrality using degree centrality. For a node v_i, the centrality encoding is a learnable vector c_i based on its degree, and is added to the node features x_i in the input layer. The formula for updating node features with centrality encoding could be represented as:

$$x_i' = x_i + c_i$$

3.4.2 Spatial Encoding

Spatial encoding involves assigning a learnable embedding based on the shortest path distance between any two nodes v_i and v_j. This is encoded as a bias term b_{ij} in the softmax attention. The spatial encoding can be described as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B\right)V$$

where B contains the bias terms b_{ij} for each node pair.

3.4.3 Edge Encoding

For edge encoding, let $SP_{ij} = (e_1, e_2, ..., e_N)$ be the shortest path from node v_i to v_j, and e_k be the feature vector of edge e_k. The edge encoding could be represented as a bias term in the attention mechanism:

$$b_{ij}^{\text{edge}} = \frac{1}{N}\sum_{k=1}^{N} e_k \cdot w_{\text{edge}}$$

where w_{edge} is a learnable embedding.

3.4.4 Graphormer Layer

The Graphormer layer uses a modified Transformer architecture. The layer formula, including multi-head self-attention (MHA) and layer normalization (LN), is:

$$h^{(l)} = \text{FFN}(\text{MHA}(\text{LN}(h^{(l-1)})) + h^{(l-1)})$$

where $h^{(l)}$ is the output of the l-th Graphormer layer, and FFN is the feed forward network, which is two linear transformation with GELU as activation function.

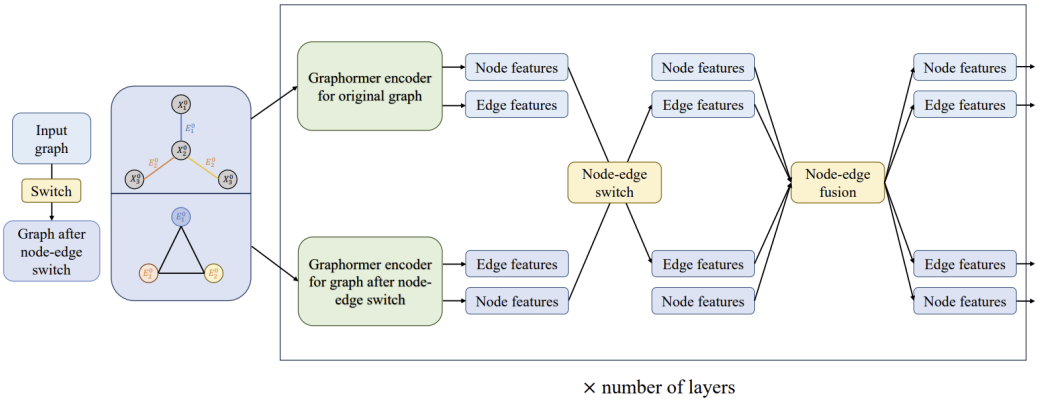

Figure 1: Model architecture

3.5 Model architecture

A brief description of our model is shown in Figure 1. For the original input with node features $X \in R^{N_V \times E_V}$, edge attribute $\in R^{2 \times N_E}$, edge features $\in R^{N_E \times D_E}$, we first build a dense matrix for edge features $E_N \in R^{N_V, N_V, D_E}$, and do the same for the graph after node edge switch. Then for each encoding layer, we feed the original graph and the graph after switch into a graphormer encoder and a GAT encoder, respectively. Then we replace the edge features in the original graph by the node features output of the switched graph. And similarly, we replace the edge features in the switched graph by the node features in the original graph. And at last, we feed the outputs of the GAT and graphormer encoder into our node edge fusion module. After 6 encoding layers, we apply a layer norm to the final output of the class token in the last graphormer encoder for the orginal graph, and use a linear transformation for the final prediction task.

4 Experiments

4.1 Dataset

In this study, we use the ogbg-molhiv dataset, which contains 41,127 graphs with an average node number of 25.5 and an average edge number of 27.5. The task is to classify each graph into two groups: whether the molecular can suppress the replication of the HIV virus or not.

4.2 Hyperparameter settings

In this study, we set the node embedding dimension into 512, and so does the dimension of FFN in the graphormer encoder. We set the dropout rate in FFN and attention mechanism into 0.1, and number of heads in multihead attention into 32. And the number of encoding layers is set to 6. No data preprocessing was performed except for node-edge switch illustrated before. We repeat our experiment 3 times with different random seeds: 114, 514, and 114514.

4.3 Comparison with other methods and ablation study

We compared our model with two most popular graph transformers: Graphormer (the version of our model without node-edge switch) and GRIT[14]. All models are trained on the ogb-molhiv dataset without pre-training, and the hyperparameters are the same as provided in their original papers. For all three models, we repeat the experiment for 3 times with different random seeds, and the results are shown in table 1:

The training/test curve was also recorded for analysis. As shown in Figure 2, we can learn that our model suffers a little from overfitting, since at the last few epoches, the training ROCAUC continues to increase while the validation and test ROCAUC decrease.

Table 1: Comparison of Model Performance on ogb-molhiv without pretraining

Model	Dualgraphormer	Graphormer	GRIT
Validate AUCROC	0.788 ± 0.05	0.732 ± 0.05	0.744 ± 0.05
Test AUCROC	0.792 ± 0.06	0.728 ± 0.05	0.748 ± 0.05

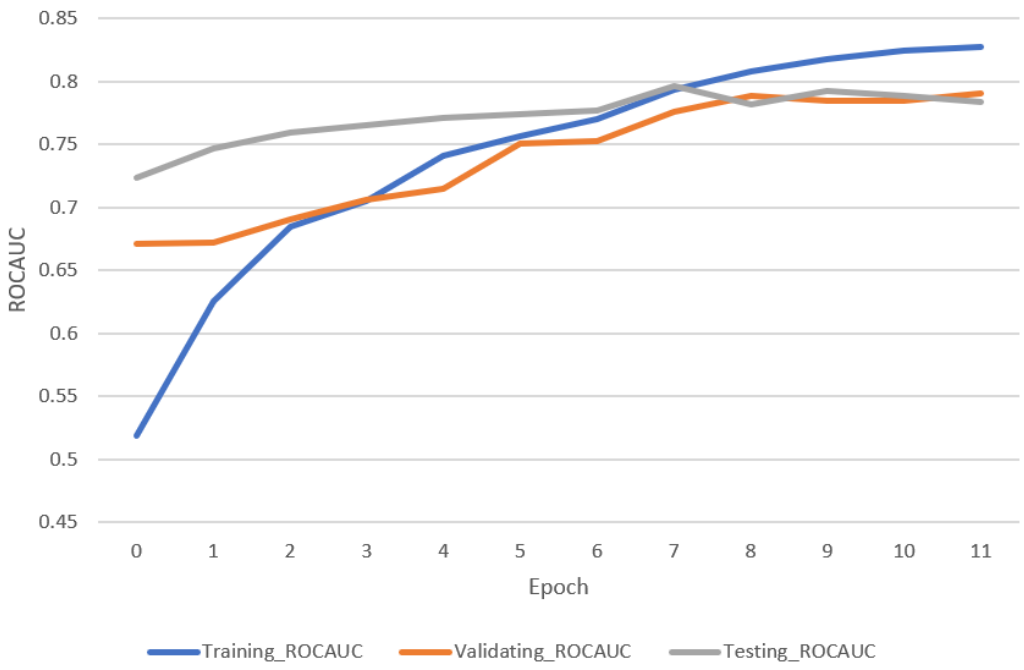

Figure 2: The change of training, validating, and testing ROCAUC of our model

5 Conclusion

Comprehensive representation of a graph require both sufficient encoding of graph structure, node features, and edge features. However, previous works paid little attention to the update of edge features. So here, we introduce DualGraphormer, a novel graph deep learning architecture for predicting molecular properties. This model significantly enhances predictive accuracy by innovatively integrating node-edge fusion and switch mechanisms, addressing the limitations of previous models. Our results demonstrate substantial improvements over existing graph transformer methods, especially for small datasets, highlighting DualGraphormer's potential in computational chemistry and drug discovery.

Acknowledgments and Disclosure of Funding

I would like to express my sincere gratitude to Professor Rex Ying and all the teaching assistants for their invaluable guidance and insights throughout this project. Their expertise and feedback were instrumental in shaping this work. Additionally, I appreciate the computing resources provided by Yale University which were essential in completing this project.

6 Supplementary Material

The code and models are publicly available at https://github.com/lugia-xiao/Dualgraphormer .

References

[1] Tong Geng, Chunshu Wu, Yongan Zhang, Cheng Tan, Chenhao Xie, Haoran You, Martin C. Herbordt, Yingyan Lin, and Ang Li. I-GCN: A Graph Convolutional Network Accelerator with Runtime Locality Enhancement through Islandization. *arXiv e-prints*, page arXiv:2203.03606, March 2022.

[2] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Bad for Graph Representation? *arXiv e-prints*, page arXiv:2106.05234, June 2021.

[3] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv e-prints*, page arXiv:2005.00687, May 2020.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv e-prints*, page arXiv:1706.03762, June 2017.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv e-prints*, page arXiv:2010.11929, October 2020.

[6] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking Graph Transformers with Spectral Attention. *arXiv e-prints*, page arXiv:2106.03893, June 2021.

[7] Tejaswini Ananthanarayana, Lipisha Chaudhary, and Ifeoma Nwogu. SignNet: Single Channel Sign Generation using Metric Embedded Learning. *arXiv e-prints*, page arXiv:2212.02848, December 2022.

[8] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv e-prints*, page arXiv:1609.02907, September 2016.

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *arXiv e-prints*, page arXiv:1710.10903, October 2017.

[10] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. *arXiv e-prints*, page arXiv:1703.06103, March 2017.

[11] Meiqi Chen, Yuan Zhang, Xiaoyu Kou, Yuntao Li, and Yan Zhang. r-GAT: Relational Graph Attention Network for Multi-Relational Graphs. *arXiv e-prints*, page arXiv:2109.05922, September 2021.

[12] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks. *arXiv e-prints*, page arXiv:2102.09844, February 2021.

[13] Xiaodong Jiang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. Co-embedding of Nodes and Edges with Graph Neural Networks. *arXiv e-prints*, page arXiv:2010.13242, October 2020.

[14] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph Inductive Biases in Transformers without Message Passing. *arXiv e-prints*, page arXiv:2305.17589, May 2023.