A Spectral Domain Graph Transformer Model with Position-Encoded Information

Baisen Xiong, Sijie Wen, Ping Lu*, Kaibiao Lin

Abstract— **The incorporation of the Laplacian Matrix Theory significantly boosts the predictive prowess of Graph Neural Networks(GNNs). This module acts as a preprocessing tool to enhance the generalization capability of spectral GNNs, and it also functions as a positional encoding mechanism to boost the expressiveness of graph transformers. However, most GNNs and graph transformer architectures didn`t not combine positional encoding with spectral domain neural networks. Without positional encoding information, spectral domain GNNs struggled to represent global graph topology effectively. Despite being grounded in Laplacian Matrix Theory, graph transformers exhibit also limited their capabilities in processing noise. To tackle the aforementioned two challenges, we introduced a novel method named SPECFORMER-PE, which integrates position-encoded spectral GNNs with laplacian positional information. SPECFORMER-PE enhances overall robustness, and refines the eigenvalue encoder and decoder by integrating scalable residual connections into spectral graph neural networks meet transformers(SPECFORMER). This model also supports stacking multiple transformer layers, thereby deepening the backbone network to effectively mitigate the risk of over-fitting. Experimental results on three graph datasets demonstrate the efficacy of our approach, showing that it outperforms existing GNNs and graph transformers in graph regression and node classification tasks. These results validate the synergy between the spectral domain filter and positional information encoding module.**

Index Terms—**Graph Neural Network, Spatial Graph Neural Network, Spectral Graph Neural Network, Graph Transformer, Position Encoded**

I. Introduction

Graph Neural Networks(GNNs), as initially presented by Scarselli et al.[1], have emerged as a key area of focus and interest within the academic community. Their utility has been amply demonstrated through a wide range of applications across diverse sectors, including social networking, recommendation systems, and financial risk management[2, 3]. The development of GNN theory in recent years has been diverged into two main branches: spatial GNNs and spectral GNNs. spectral GNNs predominantly employ the message-passing framework for graph data processing, focusing on local information learning with global learning as a secondary component[4, 5]. Spectral GNNs leverage the Graph Fourier Transform[6, 7] to transform spatial domain data into the spectral domain for filtering. This process is then followed by graph convolution, and afterwards, the Graph Fourier Inversion is applied to seamlessly complete the transition back to the spatial domain. In recent years, Graph Attention Networks (GAT) have made significant progress in the field of spatial GNNs, demonstrating exceptional performance in both local and global attention mechanisms[8]. In the realm of spectral GNNs, the Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, known as ChebyNet, effectively address the challenge of eigen-decomposition in large-scale graph structures during graph convolution operations[6]. The Semi-Supervised Classification With Graph Convolutional Networks(GCN) model subsequently streamlines the filter module by truncating the polynomial chain, enhancing its simplicity and efficiency compared to ChebyNet[9]. As the theoretical foundation of spectral domain GNNs has evolved from Graph Fourier Transform to GCN, spectral filters have been increasingly applied within the Graph Transformer domain[10].

The Transformer module is initially developed for natural language processing by Google. It has become increasingly popular in the field of GNNs in recent years. This is primarily attributed to its outstanding parallel processing capabilities and the powerful self-attention mechanism it employs. It has demonstrated exceptional effectiveness in recommendation systems[11]. The first use of a transformer in graph data is the Graph Transformer networks proposed by Yun et al.[12]. This model enhances the parallel processing and self-attention mechanisms of the transformer, improving its effectiveness on large graph datasets. Bidirectional Encoder Representations from Transformers(Bert) was subsequently developed to improve the efficiency of graph representation learning by alleviating the memory constraints associated with large-scale graphs. Bert employs a novel sampling approach to produce sub-graphs, enabling the transformer module to derive graph representations from these condensed elements[13]. However, through effectively integrates structural and positional graph data information into Transformers, encoding graph structure in transformers model(GraphiT) enables the Graph Transformer model to surpass traditional GNNs across many datasets[14]. Furthermore, Rethinking Graph Transformers with Spectral Attention (SAN) has demonstrated that incorporating positional information as a trainable module effectively highlights the benefits of Laplacian positional encoding in enhancing the encoding

Manuscript received June 19, 2024; revised March 26, 2025.

Baisen Xiong is a postgraduate of Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024, China (e-mail: ssscu@foxmail.com).

Sijie Wen is a postgraduate of Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024, China (e-mail: 1253180963@qq.com).

Ping Lu is a professor of Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024, China (e-mail: luping@xmut.edu.cn).

Kaibiao Lin is a professor of Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024, China (e-mail: 2010110706@t.xmut.edu.cn).

process[15]. In recent years, Recipe for a General, Powerful, Scalable Graph Transformer(GraphGPS) has developed a comprehensive taxonomy for positional and structural encodings and employed in GNNs, successfully integrating them into GNN layers[16]. Spectral graph neural networks meet transformers(SPECFORMER) has concurrently integrated spectral GNNs with transformers, achieving favorable results in graph classification applications.

While spectral GNNs have exerted significant influence across diverse domains, they still encounter two notable limitations.

(1) Although the Laplacian eigen-decomposition's eigenvalues can be manipulated by filters to reduce noise, the interrelationships among these eigenvalues are not fully considered.
(2) The incorporation of Laplacian matrix eigenvalues in the spectral domain to augment the expressive capacity of the global attention mechanism has not been thoroughly studied.

Furthermore, the Graph Transformer model encounters several challenges too.

(1) Currently, the majority of improvements to the Transformer architecture have centered on refining topological relationships and attention mechanisms within the spatial domain, while relatively little attention has been paid to trainable components in the spectral domain.
(2) It is still uncertain whether topological relationships and node features, which are conveyed by location and structural information, carry equal weight or if their relative importance differs across various datasets.

Previous studies indicate that position-encoded Information and spectral domain GNN have complementing impacts on their respective benefits and drawbacks. This prompts an study into the possible synergistic link between position encodings and spectrum filters, examining if they can augment each other's efficacy. Therefore, this study investigates whether Transformers enhanced with positional information can improve representational capacity of spectral GNNs. Based on SPECFORMER, we adopt the second-order Laplacian operator as the input for the encoder of the spectral domain signal, Following this, we design a SPECFORMER-PE model, which incorporate graph transformer and positional encoding data to enhance the training of the model.

The contributions of this paper are as follows:

(1) Our model ingeniously integrates positional encoding with a spectral GNN, augmenting representation by utilizing deep learning's ability to capture interconnections.
(2) We present a configurable scaled residual block, enabling the model to adapt more efficiently to node features and graph topology.
(3) Experiments in graph regression and node classification tasks demonstrate our model outperforms existing state-of-the-art methods across multiple datasets.

II. Related Work

Research on GNNs is often divided into two primary categories: spatial GNNs and spectral GNNs. Moreover, the Graph Transformer has become a prevalent methodology.

Spatial GNNs are essential for graph data analytics. Specially, GAT model efficiently uses local attention to gather information from neighboring nodes. Nonetheless, efficacy of GAT diminishes in intricate networks due to its restricted capacity for feature extraction. To address this issue, the Heterogeneous Graph Attention Network was created, improving the model's performance in complex network settings. The primary emphasis of spatial GNN research is the attention process. Nonetheless, deficiencies remain in noise reduction and the utilization of higher-order representations[17]. Conversely, spectral domain GNNs possess benefits in these areas.

As a unique approach to graph data processing, spectral GNN provides advantages in noise attenuation. Originating from graph signal theory, it is constrained by the significant computational requirements of eigen-decomposition. A notable contribution of ChebyNET is its employment of Chebyshev polynomial approximation in the eigen-decomposition stage, effectively bypassing the substantial computational load typically associated with eigen-decomposition while meeting the criteria for eigenvalue fitting. As a simplification of ChebyNET, GCN has attained commendable results across several datasets. Subsequently, GCNII outperforms conventional GCN in addressing over-smoothing issues, attributed to their investigation of network depth[18]. Training graph neural networks with 1000 layers (RevGNN-Deep) significantly increases the network's depth by an order of magnitude[19]. Giovanni et al.[20] also conducted a series of research on the breadth. Nevertheless, the majority of these models are predetermined or manually constructed in the eigen-decomposition polynomial, so forfeiting the capacity for learning. Consequently, a recent category of spectral GNNs relies on partial eigen-decomposition, enabling them to acquire more expressive representations via spectral domain filters[10]. Nevertheless, these conclusions lack sufficient strength to elucidate the connections among eigenvalues, particularly when dealing with substantial datasets.

Transformers have become pivotal in natural language processing due to their efficient handling of token relationships. Similarly, Graph Transformers are adept at capturing key characteristics in most cases. Significantly, the transformer has demonstrated efficacy with large-scale data. The Graph Transformer model has effectively transferred the self-attention and parallel processing qualities of the Transformer to GNNs. This has resulted in the creation of innovative position encoding modules, including those derived from random walks and Laplacian vectors, therefore augmenting the representational capacity of GNNs[12]. Recent research primarily emphasizes the integration of Graph Transformers with GNNs, either as a core component or as an auxiliary module. GraphiT and SAN integrate structural and positional information inside Transformers, thereby transforming position encoding into a learnable module[14, 15]. Subsequently, GraphGPS categorizes position encoding into distinct blocks to enhance simulation structure, while SAT employs GNN as a substructure extractor[21]. Most research on Graph

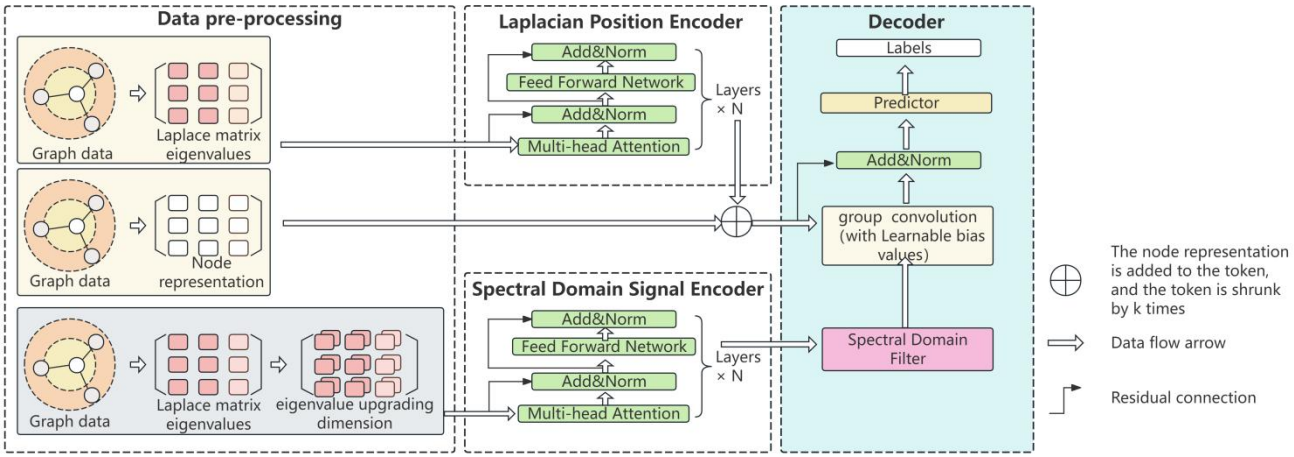

Fig. 1 The overview of SPECFORMER-PE based on spectral domain Graph Transformer and position-encoded information

Transformers focuses on using current methods to integrate graph topological relationships into node attributes while overlooking the importance of the weight between node features and topological connections.

III. Model

This section introduces the A spectral Domain Graph Transformer Model with Position-Encoded Information (SPECFORMER-PE) model, shown in Fig. 1. The model consists of three basic components: a spectral domain eigenvalue encoder, a Laplacian position encoder, and a decoder. Both encoders undertake the processing of graph data, thereby generating intermediate representations. Subsequently, the decoder integrates these intermediate representations to formulate the final representation. The encoders function as a preliminary phase for the decoder, which finally produces the model's predictions. The SPECFORMER-PE model employs a spectral domain eigenvalue encoder to extract essential eigenvalue properties and a Laplacian position encoder to ascertain optimal positional information from the second-order Laplacian matrix. This information is then fed into a decoder comprising a spectral filter, adjustable biases, and a graph convolution module. These components collectively function to produce a noise-free graph representation while preserving the global topological structure. We integrate learnable modules and unique filtering algorithms into the spectral domain filter to capture a wide range of spectral signals. This focused filtering method within the filter improves the expressive power of GNNs[22, 23]. The Graph Transformer component pertains to the concept of location encoding in GraphGPS[16]. During graph convolution, we incorporate residual connections to mitigate over-smoothing, hence improving the model's compatibility.

A. Spectral domain Signal Encoder

This module focuses on encoding the spectral domain graph signal, which is derived from the Laplacian matrix's eigen-decomposition. The graph signal serves as input, and its hidden layer representation is produced through Transformer coding, incorporating the structural information of the spectral domain signal into the coding process. The structure encoding of spectral domain graph signals is similar to the absolute position encoding module in the Transformer module, and the idea is derived from the SPECFORMER model. This module acts as a powerful filter to obtain different differences for each feature. Let $PE_{(\lambda,2i)} \in R^{n\times d_{model}}$ map each eigenvalue so that the scalars are converted into corresponding vectors.

$$PE_{(\lambda,2i)} = sin(100\lambda/10000^{2i/d_{model}}) \quad (1)$$

$$PE_{(\lambda,2i+1)} = cos(100\lambda/10000^{2i/d_{model}}) \quad (2)$$

Where i denotes the dimension of the location encoding corresponding to the graph model. The benefits of this module are as follows: (1) This module can represent the absolute position information of the eigenvalue and lift the dimension. (2) its wavelength is from 2π to $10000\cdot 2\pi$ so that the value can be quantized in a certain scale. There is a note that the absolute position information of the Transformer is similar to that of this module, but the meaning is different. The former represents the position information of words or pixels within the overall structure, generally capturing the topological relationships of discrete values across the entire dataset. In contrast, the latter represents the relationships between eigenvalues in the spectral domain. Therefore, the expression is the topological relationship of a continuous value in the whole. The above step can be seen as a dimensionality increase of the graph signal, and the trend is added after the feature dimension. After the upscaling, it will be fed into the Feed-Forward network for encoding, let $H = [\lambda_1||PE(\lambda_1), \cdots, \lambda_n||PE(\lambda_n)]^T \in R^{n\times d_{model}}$. Then, H first passes through the normalization layer and then enters the multi-head attention layer and the Feed-Forward neural network layer to calculate the self-attention weights. After the above network structure processing, the encoded spectral domain graph signal data is obtained.

$$\widehat{H} = MHA(LN(H)) + H \quad (3)$$

$$H = FFN(LN(\widehat{H})) + \widehat{H} \quad (4)$$

B. Laplacian position encoder

Position encodings are crucial for representing a node's spatial location within a network, ensuring that analogous nodes possess similar positional encoding. A common method for graph location encoding is calculating the distances of feature vectors between pairs of nodes.

Nevertheless, the drawback of this technique is the substantial computational effort needed to calculate all eigenvector distances between the nodes. In graph position encoding, our objective is to encapsulate the fundamental nature of the graph's edges. A proficient strategy is to customize the position encoding to serve this objective. We utilize the Laplacian position encoding approach from GraphGPS, which provides a partial answer to the generalization difficulty. Laplacian position encoding enhances the backbone network's ability to identify relative node locations. It is computationally efficient and very appropriate for extensive graph datasets. In conventional GNNs, the Laplacian matrix functions as an indicator of node connection and the overall topology of the network. The Laplacian matrix facilitates graph convolution by allowing the extraction of spectral characteristics via eigen-decomposition, which is essential for graph classification and regression problems.

Upon inputting the graph data, the initial steps involve decomposing the node features and normalizing the Laplacian matrix. In this case, batch normalization is used, although layer normalization is employed in other transformer modules. This provides the model with a computational time advantage. The model next utilizes the output from the preceding stage as input for the Feed-Forward neural network, including learnable features into the Laplacian matrix. Ultimately, we encode the Transformer module, which we might designate as the BERT model. This enhances the simulation of edge characteristics in graph convolution, facilitating the acquisition of relative positional information among nodes. The transformer comprises a self-attention mechanism, a multi-head attention mechanism, a Feed-Forward network, and a residual connection. For instance, Equations (5) through (8).

$$Q = W^Q H_l, K = W^K H_l, V = W^V H_l \tag{5}$$

$$Z_{PE} = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

$$\widehat{Z_{PE}} = MHA(LN(Z_{PE})) + H_l \tag{7}$$

$$\widetilde{Z_{PE}} = FFN(LN(\widehat{Z_{PE}})) + \widehat{Z_{PE}} \tag{8}$$

The above is the work of a Transformer block. This Laplacian encoder has l such Transformer blocks, where l is a hyperparameter. Stacking multiple Transformers improves the expressive power of the model and can capture the graph topology more accurately.

C. Decoder

The primary functions of this module, as delineated by the aforementioned encoder, are as follows: (1) Enhancing the expressive capacity of the graph signal via Multi-Head Attention and Feed-Forward Networks is crucial for capturing complex graph relationships, (2) individually filtering each spectral domain feature using a spectral filter, (3) concatenating the positional encoding, spectral domain graph signal, and node features while assigning their respective weights, (4) producing the final representation through GNN message passing.

1) Spectral domain Filter

The model utilizes a combination of Multi-Head Attentionand Feed-Forward Networks to include a wide spectrum of signal frequencies in the graph space prior to filtering. The model successively integrates Multi-Head Attention and Feed-Forward Networks, employing residual connections and a dropout technique to promote varied frequency representation in the graph signal. This representation is subsequently channeled through the filter, with each Multi-Head Attentionhead handled independently, so augmenting the model's capacity to record a broad range of frequencies.

$$H_j = Attention(QW_j^Q, KW_j^K, VW_j^V) \tag{9}$$

$$\lambda_j = RELU(H_j W_\lambda) \tag{10}$$

Where j denotes the number of multiple heads and H_j represents the input of the $j-th$ head, λ_j can be considered as the $j-th$ eigenvalues of the filter.

2) Learnable bias values

After the above steps the model will generate j intermediate representations, the model uses Feed-Forward Networks to redistribute the weights, and then concatenate them through a final dimension to make it learnable in the process. In this process, methods related to the graph Fourier transform are used.

$$Z_j = Udaig(\lambda_j)U^T \tag{11}$$

$$\widehat{Z}_j = FFN(I||Z_1||\cdots||Z_j) \tag{12}$$

Here, U is the eigenvector corresponding to the spectral domain eigenvalue and represents the spatial representation generated by the graph Fourier transform of the $j-th$ head, $Z_j\widehat{Z}_j \in R^{n\times d_{model}}$.

3) Graph convolution module

In this module, we aim to aggregate the hidden layer representations produced in the above modules into the final representation for the full connection. First, the representations $Z^1_{node+pe}$ generated by the above modules need to be combined. To achieve this, the model introduces a coefficient into the residual network. Currently, this coefficient is set to a constant value, k, which is studied as a hyperparameter. Then, The bias value $\widehat{Z}_j$ generated above through the graph convolutional layer and the residual network $Z^{l-1}_{node+pe}$ are used for the Hadamard product. Then, it is entered into the Feed-Forward network to learn the representation. Equations (14) and (15) compose a graph convolutional layer, and the graph convolutional layer can be stacked many times in the module. Finally, if the task involves node classification, the output will pass through a fully connected layer and then undergo a softmax operation to obtain the predicted probabilities. These probabilities are then used to compute the cross-entropy loss. If the graph regression task is carried out, there is an average pooling layer before the full connection, and the binary classification loss function or L_1 loss function is entered.

$$Z^1_{node+pe} = \widetilde{Z_{PE}} + k \cdot Z_{node} \tag{13}$$

$$Z^{l-1}_{node} = \widehat{Z}_j * Z^{l-1}_{node+pe} \tag{14}$$

$$\widetilde{Z^l_{node}} = RELU(Z^{l-1}_{node} * W^{l-1}_{node}) + \widetilde{Z^{l-1}_{node}} \tag{15}$$

IV. Theoretical analysis

This work examines spectral filtering and positional encoding; nevertheless, the integration of positional encoding in the spatial domain with filtering in the spectral domain is paradoxical. This work aims to provide a

theoretical explanation by referencing the relationship between the spatial domain and the spectral domain as outlined by Chen et al.[24]. Corollary 1: Local aggregation in the spatial domain corresponds to modifying weights among signals of varying frequencies in the spectral domain, whereas neighbors of differing orders in the spatial domain equate to signal components of distinct frequencies in the spectral domain. Moreover, the position-coding in this manuscript might be seen as an ancillary variable in the spatial domain or a deviation metric for the filter.

Brief proof Corollary 1 Take GCN as an example:

$$Z = \varphi X + \psi D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X = (\varphi I + \psi D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X \quad (16)$$

$$Z = (\varphi I + \psi \widetilde{A}) X \quad (17)$$

Equation (16) is the update process of GNN node embedding in the spatial domain, where X represents the node characteristics, φ, ψ represents the weight, D represents the degree matrix, A represents the adjacency matrix $\widetilde{A}$, and represents the normalized adjacency matrix. It can be seen that Equation (16) is simplified to Equation (17). When φ=0, ψ=1, the GCN Equation (18) can be obtained, with a slight difference in the degree matrix $\widehat{D}$ and the adjacency matrix $\widehat{A}$ of the self-loop added.

$$Z = \widetilde{A} X = \widehat{D}^{-\frac{1}{2}} \widehat{A} \widehat{D}^{-\frac{1}{2}} X \quad (18)$$

A lot of work in the spectral domain of GNNs can be understood as adjusting the weights of frequency components during the aggregation process. In other words, the frequency filter g is applied:

$$Z = (\sum_{i=0}^{l} \theta_i \lambda_i u_i u_i^T) X = U g_\theta(\Lambda) U^T X \quad (19)$$

Equation (19) represents a process of reversing the spatial domain to the spectral domain and then to the spatial domain. The first half Equation u_i represents the feature vector of the I-node graph signal, λ_i represents the eigenvalue, θ represents the filter function, and X represents the node feature. The second half Equation represents the matrix form, U represents the set of eigenvectors, the diagonal matrix Λ is composed of eigenvalues, and the filter function g_θ parameterized by θ. For GCN, Equation (20) is the spectral domain interpretation of GCN and is a low-pass filter, that is, smaller eigenvalues can pass, and the relationship is explained $G(\Lambda) = 1 - \Lambda$.

$$Z = \widetilde{A} X = (I - \tilde{L}) X = U(1 - \Lambda) U^T X \quad (20)$$

When considering the process of high-order neighbors, g_θ is expressed as a polynomial, and the spatial normalization matrix $\widetilde{A}$ is also more complex. The specific proof process can be referred to the original paper.

High-order aggregation in spatial domain:

$$Z = (\varphi I + \sum_{j=1}^{k} \psi_i (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})^j) X = P(\widetilde{A}) X \quad (21)$$

spectral domain high-order aggregation:

$$Z = (\sum_{i=0}^{l} \sum_{j=0}^{k} \theta_i \lambda_i^j u_i u_i^T) X = U P_\theta(\Lambda) U^T X \quad (22)$$

Where P, P_θ is a polynomial function.

V. Experiment

In this section, we perform comparative experiments on three datasets. In the node-level task and graph regression task respectively, our model is the state-of-the-art results (hereafter referred to as SOTA) in most cases.

A. Datasets and Settings

Our model validation work is mainly performed on two tasks, node classification and graph-level regression tasks. ZINC is a molecular dataset containing 12000 molecules corresponding to a graph regression task. Both PATTERN and CLUSTER are challenging node classification tasks. PATTERN contains 10,000 graphs with an average of 118.9 nodes, and CLUSTER contains 10,000 graphs with an average of 117.2 nodes[22].

For hyperparameters, we set different settings for hidden layer dimensions, number of attention heads, and number of Transformer layers to make the model more relevant to the dataset. ZINC uses 128 hidden dimensions, CLUSTER uses 48 hidden dimensions, and PATTERN uses 16 hidden dimensions. For the number of heads and layers, ZINC has 8 layers with 16 heads, CLUSTER has 8 layers with 4 heads, and PATTERN has 1 layer with 4 heads, respectively. The Adam optimizer with warm-up was used for training, and the model reduced the learning rate with the increase of epochs to improve the performance. For ZINC, since it is a graph regression task, the model only uses the l1 foss function to calculate the mean absolute error. To perform node classification tasks, both CLUSTER and PATTERN models utilize the cross-entropy loss function. Subsequently, they employ a confusion matrix to evaluate accuracy.

TABLE I
COMPARISON OF OUR MODEL WITH THE SOTA MODEL IN GRAPH REGRESSION TASK AND NODE CLASSIFICATION TASK

Methods	ZINC↓ MAE	CLUSTER↑ ACC	PATTERN↑ ACC
GCN	0.367± 0.011	68.498± 0.976	71.892± 0.334
GIN	0.526± 0.051	64.716± 1.553	85.387± 0.136
Graph Transformer-sparse	0.226± 0.014	73.169± 0.622	84.808± 0.068
Graph Transformer-full	0.598± 0.049	27.121± 8.471	56.482± 3.549
SAN-sparse	0.198± 0.004	75.738± 0.106	81.329± 2.150
SAN-full	0.139± 0.006	76.691± 0.247	86.581± 0.037
Graphormer	0.122± 0.006	-	-
SAT	0.094± 0.008	77.856± 0.104	86.865± 0.043
GraphGPS	0.070± 0.004	78.016± 0.180	86.685± 0.059
SPECFORMER	0.066± 0.003	-	-
SPECFORMER-PE (ours)	0.064± 0.005	78.711± 0.034	90.822± 0.055

B. Baseline

We will examine many prominent GNN and Graph Transformer models, including the latest SOTA model. The GNN baselines comprise GCN and GIN. GCN is a traditional convolutional model in GNNs that serves as a benchmark across several domains, whereas GIN employs the WL-test to accommodate graph topology for graph isomorphism networks. Graph Transformer methodologies encompass Graph Transformer[12], SAN[15], Graphormer[25], SAT[21], and SPECFORMER[10]. The

Graph Transformer innovates the utilization of Transformers for graph architectures, demonstrating potential on sparse datasets. SAN improves spectral domain attention by using learnable location data and Transformers. Graphormer provides three types of encodings-centrality, spatial, and edge position to correspond with graph topologies. SAT, a structure-aware Transformer methodology, has exhibited robust performance recently. SPECFORMER, by incorporating spectrum filters into its Graph Transformer technology, has achieved success across several datasets.

C. *Main Results*

TABLE I encapsulates the experimental outcomes of our model concerning graph regression and node categorization, using data from various baseline models received from the research. Our model exceeds the leading methodologies by around 1% in accuracy on both the CLUSTER and PATTERN datasets, and by about 0.002 in mean absolute error on the ZINC dataset. This result illustrates that the Graph Transformer significantly improves both global and local attention by using the Laplacian filter with Laplacian position encoding, especially in node classification tasks.

D. *Hyperparameter Study*

Previous research on spectral domain GNNs indicated that each eigenvalue of the Laplacian matrix corresponds to a class of frequency graph signals, suggesting potential relationships among graph signals. When integrated with the prior distinct filtering technique, it may be posited that each head in the multi-head attention represents a synthesis of various frequency graph signals, and the optimal number of graph signal types for achieving the optimum results remains undetermined. Consequently, we do a hyperparameter analysis on the ZINC dataset pertaining to multiple heads. ZINC was chosen because the assessment metric for this dataset is the average variance error, which explicitly indicates the discrepancy between the anticipated and actual values. The quantity of heads must obscure the common divisor of the layer dimension, impacting video memory; so, more traditional head counts are used, as seen in Fig. 3 below. In the scenario of 128 dimensions, selecting 16 heads yields optimal results for both the test and training datasets, with outcomes of 0.0935 and 0.0640, respectively.

We previously examined the hyperparameter of the number of heads, which indicates the extent of graph signal representation in spectral domain neural networks. Another aspect worthy of examination is the residual connection associated with the parameter. The value of k delineates the correlation between point characteristics and graph topology throughout the graph convolution process. Extensive experimentation with this hyperparameter has revealed that its value has a nonlinear feature, as illustrated in Fig. 2 below.

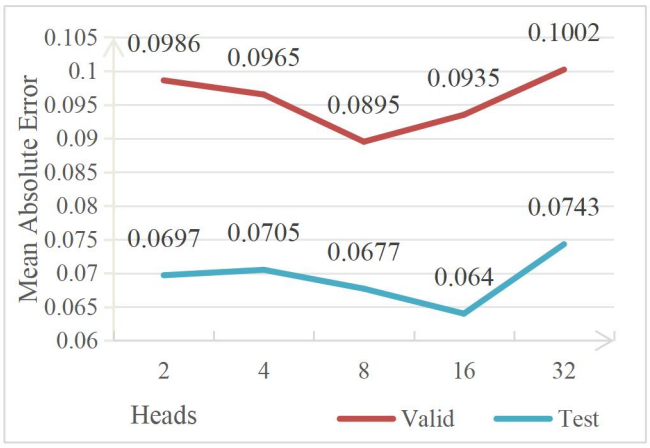

Fig. 2 Multi-head attention experiments on zinc data sets

Furthermore, we conduct hyperparameter studies on the CLUSTER and PATTERN datasets. The experiment demonstrates that the diversity of layers across various data sets results in fluctuations in accuracy. Fig 4 and 5 illustrate that CLUSTER exhibits superior performance at an elevated level, but PATTERN demonstrates inferior performance at a lower level. In the PATTERN dataset, over-smoothing manifests at layer 2 and above, with an accuracy of 82.378%.

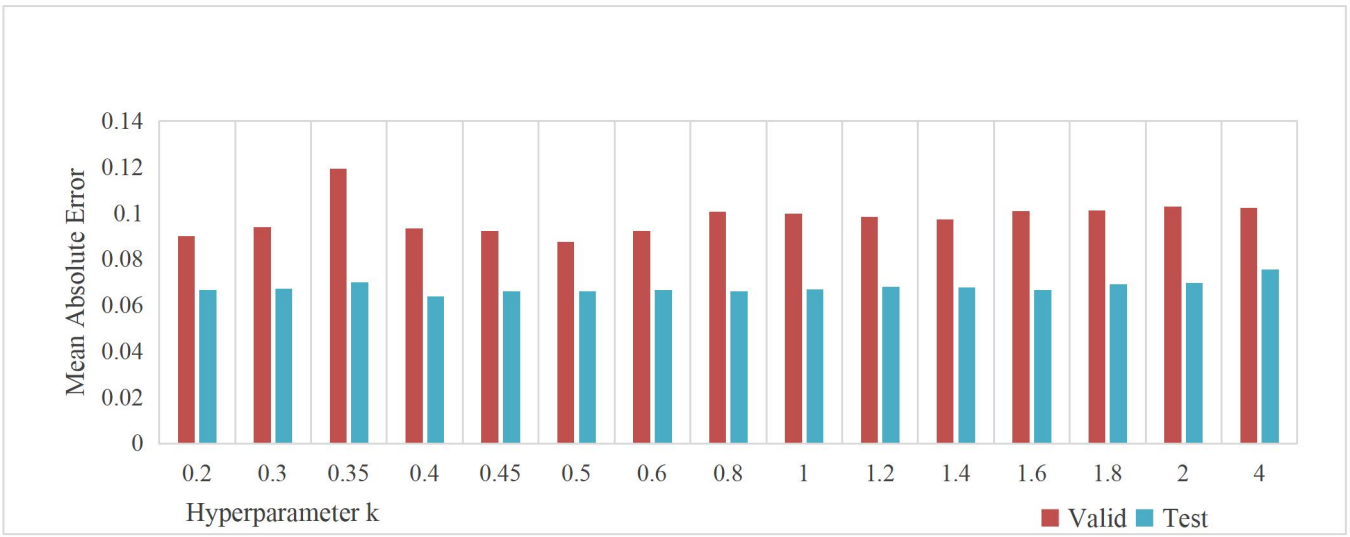

Fig. 3 Experiments on hyperparameter k values in zinc data sets

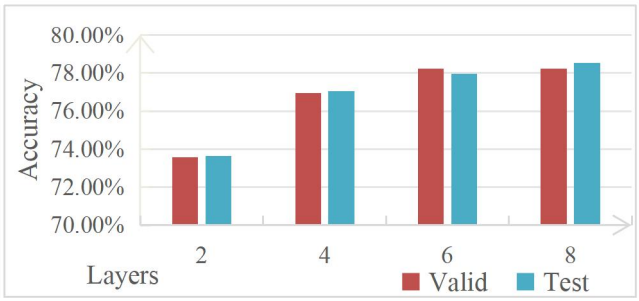

Fig. 4 The accuracy of CLUSTER varies with layers

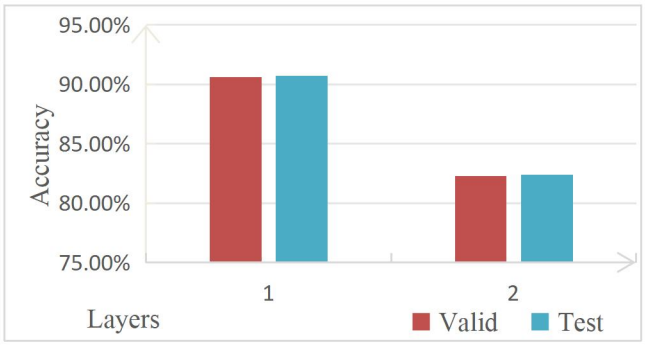

Fig. 5 The accuracy of PATTERN varies with layers

E. Ablation studies

This study presents ablation experiments on the CLUSTER and PATTERN datasets for node classification tasks, where the Laplacian coding module and the scaled residual connection are individually ablated to assess the impact of each module on the model. TABLE II illustrates the significance of the Laplacian coding module for the model, with the use of scaled residual connections enhancing its accuracy. The SPECFORMER model has not been tested experiments on the CLUSTER and PATTERN datasets. We adapt the associated embedding module of SPECFORMER and incorporate softmax and cross-entropy loss functions for evaluation. The impact of the Laplacian module is more pronounced in the CLUSTER dataset, and there is a little enhancement in the residual connection.

TABLE II
COMPARE THE EFFECT OF EACH MODULE IN CLUSTER AND PATTERN

	CLUSTER		PATTERN	
	ACC Valid	ACC Test	ACC Valid	ACC Test
SPECFORMER	77.843± 0.311	77.924± 0.242	89.753± 0.102	89.837± 0.204
SPECFORMER-PE	78.342± 0.231	78.623± 0.123	90.621± 0.342	90.771± 0.122
SPECFORMER-PE+ residual connection	78.893± 0.151	78.711± 0.134	90.703± 0.101	90.822± 0.131

VI. DISCUSSION

Comparison of GNN models: The fundamental concept of GNN is to represent the interconnections among nodes inside a graph structure, which constitutes both its advantage and its limitation. The advantages of GNN lie in its requires computing nodes individually, hence facilitating the integration of global information and ensuring the model's invariance, equivariance, and robustness in learningthe spectral domain filter and the Laplacian position encoding within extensive graph structures. The drawback is that GNN must process each node sequentially, resulting in serial calculations on extensive graph topologies, which is detrimental. Consequently, in comparison to GNN, the primary attribute of our model is its parallel computing capability derived from the Transformer architecture. Additionally, the self-attention mechanism enhances focus on the local neural network, while the Laplacian positional encoding demonstrates effective performance in global attention.

Comparison with Graph Transformer model: The fundamental concept of Graph Transformer is to explore whether the edge relationships within a graph structure can be captured by a nonlinear function. In this framework, each node is treated as part of a fully connected graph, allowing it to exchange messages with other nodes. This approach currently incorporates elements of spatial GNN; nonetheless, it exhibits inadequate graph representation capabilities, and noise inside the spatial domain adversely impacts the resulting representation. Our model has been partially examined in the spectrum domain by mitigating some noise effects using filters and integrating Laplacian positional encoding. The spectral domain filter and the Laplacian positional encoding, both derived from the properties of the Laplacian matrix, are likely interconnected. The utilization of the Transformer block to model this connection is commendable.

VII. CONCLUSIONS

This paper proposes a novel integration of spectral domain filters with Laplacian positional encoding, emphasizing node classification when Laplacian encoding is essential. Our model utilizes dual encoding modules to highlight the benefits of Laplacian-based methods in graph problems. The exceptional performance across several datasets highlights the pedagogical significance of integrating Graph Transformer with spectral GNN methodologies.

Limitations: Our model employs a Laplacian matrix for eigen-decomposition, necessitating substantial computational resources to compute the eigenvalues of the Laplacian matrix. Consequently, for a singular network structure with extensive nodes, this level of processing is catastrophic. Furthermore, the model requires extensive training time because of the computational intricacies associated with the two Transformer blocks and the second-order Laplacian operator, namely self-attention and the aggregation of multi-layer networks.

Future work: Given the limitations mentioned above, we can consider adding a lightweight module. In addition, there is an important question of whether we can go further in-depth so that the Transformer can simulate the topological relationship on the graph structure more comprehensively. Finally, this model may have good performance in the field of molecular graphs, and it has good computing power and learning ability based on the datasets with multiple graphs but a small average number of nodes.

REFERENCES

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. J. I. t. o. n. n. Monfardini, "The graph neural network model," IEEE transactions on neural networks, vol. 20, no. 1, pp. 61-80, 2008.

[2] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," Neurocomputing, vol. 549, p. 126441, 2023.

[3] L. Yang et al., "Dgrec: Graph neural network for recommendation with diversified embedding generation," in Proceedings of the sixteenth ACM international conference on web search and data mining, 2023, pp. 661-669.

[4] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261, 2018.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in International conference on machine learning, 2017, pp. 1263-1272.

[6] M. Defferrard, X. Bresson, and P. J. A. i. n. i. p. s. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," Advances in neural information processing systems, vol. 29, 2016.

[7] J. Bruna, W. Zaremba, A. Szlam, and Y. J. a. p. a. LeCun, "Spectral networks and locally connected networks on graphs," arXiv preprint arXiv:1312.6203, 2013.

[8] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. J. s. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, vol. 1050, no. 20, pp. 10-48550, 2017.

[9] T. N. Kipf and M. J. a. p. a. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[10] D. Bo, C. Shi, L. Wang, and R. J. a. p. a. Liao, "Specformer: Spectral graph neural networks meet transformers," arXiv preprint arXiv:2303.01028, 2023.

[11] Q. Wang and W. Zhang, "Session-based Recommendation Algorithm Based on Heterogeneous Graph Transformer," IAENG International Journal of Computer Science, vol. 50, no. 4,pp1347-1353, 2023.

[12] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. J. A. i. n. i. p. s. Kim, "Graph transformer networks," Advances in neural information processing systems, vol. 32, 2019.

[13] J. Zhang, H. Zhang, C. Xia, and L. J. a. p. a. Sun, "Graph-bert: Only attention is needed for learning graph representations," arXiv preprint arXiv:2001.05140, 2020.

[14] G. Mialon, D. Chen, M. Selosse, and J. J. a. p. a. Mairal, "Graphit: Encoding graph structure in transformers," arXiv preprint arXiv:2106.05667, 2021.

[15] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. J. A. i. N. I. P. S. Tossou, "Rethinking graph transformers with spectral attention," arXiv preprint arXiv:1806.01261, 2021.

[16] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. J. A. i. N. I. P. S. Beaini, "Recipe for a general, powerful, scalable graph transformer," Advances in Neural Information Processing Systems, vol. 35, pp. 14501-14515, 2022.

[17] X. Wang et al., "Heterogeneous graph attention network," in The world wide web conference, 2019, pp. 2022-2032.

[18] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in International conference on machine learning, 2020, pp. 1725-1735.

[19] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in International conference on machine learning, 2021, pp. 6437-6449.

[20] F. Di Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lio, and M. M. Bronstein, "On over-squashing in message passing neural networks: The impact of width, depth, and topology," in International Conference on Machine Learning, 2023, pp. 7865-7885.

[21] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in International Conference on Machine Learning, 2022, pp. 3469-3489.

[22] [M. Yang, Y. Shen, R. Li, H. Qi, Q. Zhang, and B. Yin, "A new perspective on the effects of spectrum in graph neural networks," in International Conference on Machine Learning, 2022, pp. 25261-25279.

[23] X. Wang and M. Zhang, "How powerful are spectral graph neural networks," in International Conference on Machine Learning, 2022, pp. 23341-23362.

[24] Z. Chen et al., "Bridging the gap between spatial and spectral domains: A survey on graph neural networks," arXiv preprint arXiv:2002.11867, 2020.

[25] C. Ying et al., "Do Transformers Really Perform Bad for Graph Representation?(2021)," arXiv preprint arXiv:2106.05234., 2021.