Hierarchical Diffusion Scattering Graph Neural Network

Ke Zhang , Xinyan Pu , Jiaxing Li , Jiasong Wu , Huazhong Shu , Youyong Kong[*]
Jiangsu Provincal Joint International Research Laboratory of Medical Information Processing,
School of Computer Science and Engineering, Southeast University, Nanjing, China
{kylenz, 220201976, jiaxing_li, jswu, shu.list, kongyouyong}@seu.edu.cn

Abstract

Graph neural network (GNN) is popular now to solve the tasks in non-Euclidean space and most of them learn deep embeddings by aggregating the neighboring nodes. However, these methods are prone to some problems such as over-smoothing because of the single-scale perspective field and the nature of low-pass filter. To address these limitations, we introduce diffusion scattering network (DSN) to exploit high-order patterns. With observing the complementary relationship between multi-layer GNN and DSN, we propose Hierarchical Diffusion Scattering Graph Neural Network (HDS-GNN[1]) to efficiently bridge DSN and GNN layer by layer to supplement GNN with multi-scale information and band-pass signals. Our model extracts node-level scattering representations by intercepting the low-pass filtering, and adaptively tunes the different scales to regularize multi-scale information. Then we apply hierarchical representation enhancement to improve GNN with the scattering features. We benchmark our model on nine real-world networks on the transductive semi-supervised node classification task. The experimental results demonstrate the effectiveness of our method.

1 Introduction

As the generalization of CNN to non-Euclidean space, Graph Neural Networks (GNNs) have been widely studied in recent years. Starting from the success of GCN on semi-supervised node-level task [Kipf and Welling, 2017], various efficient graph-based models have been proposed to solve graph-related tasks, such as link prediction [Zhang and Chen, 2018], node classification [Veličković *et al.*, 2018], graph classification [Lee *et al.*, 2019]. Based on these fundamental tasks, GNNs are widely applied in various fields, such as traffic forecasting [Jiang and Luo, 2021] and neuroscience [Kong *et al.*, 2021].

[*]Contact Author

[1]The codes are available at https://github.com/Anfankus/hds-gnn

Most existing GNNs encode graph representation in message passing criteria, i.e. propagating and aggregating neighboring nodes to learn deep representations of the central node [Kipf and Welling, 2017; Hamilton *et al.*, 2017; Veličković *et al.*, 2018]. However, recent studies reveal that GNN acts as actually a low-pass filter on graph signal [Li *et al.*, 2018; Nt and Maehara, 2019; Balcilar *et al.*, 2021a], which makes nodes undistinguishable when stacking multiple layers, and leads to oversmoothing problem. Additionally, GNNs usually perform on a fixed range of neighbors and directly aggregate the shallow representations of the neighboring nodes. This kind of approaches lack multi-scale information for learning intrinsic properties and is limited to a local receptive field, which prevents information propagation over long distances.

To overcome the weaknesses mentioned above, we introduce diffusion scattering network (DSN) [Gama *et al.*, 2018] to provide multi-scale band-pass signals for GNNs. DSN is a variant of scattering network [Mallat, 2012] that utilizes diffusion wavelet [Coifman and Maggioni, 2006] to perform multi-scale analysis and build stable representations of graph signals. Some recent works have studied the application of scattering networks to graph tasks. [Gao *et al.*, 2019; Zou and Lerman, 2020] aggregates the wavelet coefficients built from all scattering paths to just one representation. [Gama *et al.*, 2018] builds a diffusion GNN that only applies the first layer of scattering network as the convolution operator, and each layer of the model corresponds to a different scattering scale. And [Min *et al.*, 2020; Min *et al.*, 2021] only select the scattering features which are built through several pre-defined paths to provide fragmented band-pass signals. Most of these works rely on the handcrafted scattering features and none of them make use of the hierarchical properties of the scattering network, which is also important for a distinguishable representation.

Our method is elicited by an interesting observation between multi-layer GNN and DSN, that is the deeper layers of GNN continually smooth the signals while the deeper layers of DSN extract finer signals. And this observation naturally leads to a hierarchical fusing approach. In this work, we propose a novel graph learning model named Hierarchical Diffusion Scattering Graph Neural Network (HDS-GNN) to build the adaptive node-level scattering features, and bridge DSN and GNN by fusing representations layer by layer. Additionally, we adaptively weigh the different scales to strengthen the useful scales and reduce the impact of noise.

In summary, our contributions in this work are:

- By intercepting the low-pass filtering in the scattering network, we extract node-level information from DSN, which also contains multi-scale band-pass signals. And we apply *inter-scale weighting* to adaptively regularize the different scales.
- We propose a new framework to bridge DSN and GNN, named Hierarchical Diffusion Scattering Graph Neural Network (HDS-GNN). HDS-GNN supplements the backbone GNN with scattering features layer by layer to enhance graph representation. This approach effectively exploits the complementary properties of DSN and GNN layer by layer.
- Our model is evaluated on nine real-world networks. The experimental results demonstrate the effectiveness of our model, as well as a significant improvement over the backbone GNNs.

2 Background

Note. When we consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$, it usually contains three elements: nodes, edges and edge weights. Every node $v_i \in \mathcal{V}$ has a d-dimension feature x_i and $X \in \mathbb{R}^{n\times d}$ is the feature matrix of all nodes. Every edge $(v_i, v_j) \in \mathcal{E}$ where $i \neq j$ is an undirected edge connecting v_i and v_j with an edge weight $\omega_{ij} \in \mathcal{W}$. Also, the connectivity of $\mathcal{G}$ can be represented by an adjacency matrix A or a weighted adjacency matrix W. For node-level task, every node has a class label $y_i \in Y$; for graph-lavel task, every graph has a class label $\mathcal{Y}$.

2.1 Graph Neural Network

Graph neural network is well-studied recently to embed graph structured data. Most current GNNs [Kipf and Welling, 2017; Veličković *et al.*, 2018; Hamilton *et al.*, 2017] follow message passing mechanism which aggregates the local neighborhoods and updates the target node:

$$H_v^{(l+1)} = upd(H_v^{(l)}, agg(\{H_u^{(l)}, u \in \mathcal{N}(v)\}) \qquad (1)$$

where H_v denotes the representation of node v; $agg(...)$ aggregates the neighbor nodes $\mathcal{N}(v)$, which can be replaced by any neighbor sampling method, and outputs a *message*; $upd(v, message)$ updates H_v according to the *message* and v itself. For example, GCN [2017] set $agg = \sum_u 1/\sqrt{(d_u+1)(d_v+1)} H_u^{(l)}$ and $upd = (1/(d_v + 1)H_v^{(l)} + agg)\Theta^{(l)}$; GAT [2018] learns coefficients in agg based on attention mechanism.

2.2 Diffusion Scattering Network

The scattering transform [Mallat, 2012; Bruna and Mallat, 2013] is a multi-resolution analysis method to build stable representations of images (such as translation invariance and rotation invariance) with wavelets. And the diffusion scattering network generalizes it to the non-Euclidean domain by leveraging graph diffusion wavelet.

The graph diffusion wavelet is defined by the power of diffusion operator to accomplish multi-resolution analysis. The

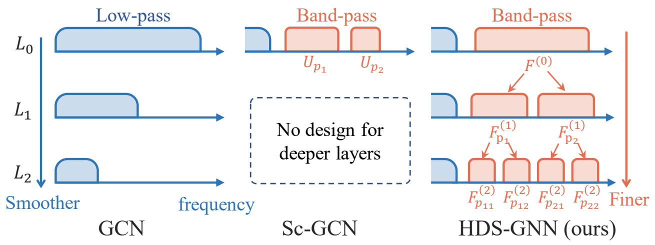

Figure 1: Comparison in frequency domain

symmetric diffusion operator defined in [Gama *et al.*, 2018] is: $T_{sym} = \frac{1}{2}(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$; and the random walk operator defined in [Gao *et al.*, 2019] is: $T_{rw} = \frac{1}{2}(I + WD^{-1})$, where both D denote the digonal degree matrix. With the multiple powers of the diffusion operator, a series of multi-scale wavelet filters can be constructed according to [Coifman and Maggioni, 2006] :

$$\psi_0 = I - T, \psi_j = T^{2^{j-1}}(I - T^{2^{j-1}}) = T^{2^{j-1}} - T^{2^j} \qquad (2)$$

where j denotes orders. Then, we can build a filter bank $\mathbf{\Psi} = \{\psi_0, \ldots, \psi_J\}$ with both spatial and spectral localization. The diffusion wavelet coefficients are obtained with the filter bank: $\mathbf{\Psi}_J(G, x) = \{\psi_0 x, \ldots, \psi_J x\}$. The diffusion scattering transform $\Phi_{J,L}(G, x)$ is constructed by cascading the wavelet operator $\mathbf{\Psi}$, a point-wise nonlinearity operator ρ and a low-pass operator U, where L denotes layers and J denotes orders. The scattering representation of each layer in the scattering network is:

$$\begin{cases} \phi_0 = Ux, \\ \phi_{1J} = U\rho\mathbf{\Psi}_J(G, x) = U\rho\mathbf{\Psi}x \\ \phi_{lJ} = U\rho\mathbf{\Psi}\ldots\rho\mathbf{\Psi}x = U(\rho\mathbf{\Psi})^l x, 0 \leq l < L \end{cases} \qquad (3)$$

3 Graph Signal Processing in GNN and DSN

Due to the aggregation of direct neighbors, GNNs actually act as a low-pass filter on spectral [Balcilar *et al.*, 2021a; Nt and Maehara, 2019; Li *et al.*, 2018] which smoothes the graph signal, and the smoothness is also considered as the key to success of GNNs [Balcilar *et al.*, 2021b]. This success is based on a premise that adjacent nodes are similar, which is not completely feasible for some real-world networks, e.g., nodes of different classes connected together [Xu *et al.*, 2019]. This kind of connectivity corresponds to high-frequency signals in the spectral domain, and reasonable use of these signals can be an effective way to enhance GNNs. In addition, as GNN going deeper, the signal is smoother. As illustrated in Figure 1, the low-pass band of GCN is narrowed layer by layer. This leads to a worse ability to distinguish nodes and eventually leads to oversmoothing.

As introduced above, graph scattering network extracts multi-scale band-pass signals by cascading wavelet operators and is able to provide high frequency signals that GNN lacks. Recent works [Min *et al.*, 2020; Min *et al.*, 2021] extract scattering features with several pre-defined scattering path, resulting in fixed-width band-pass signals, as shown in Figure 1(b). These methods only add fixed scattering features

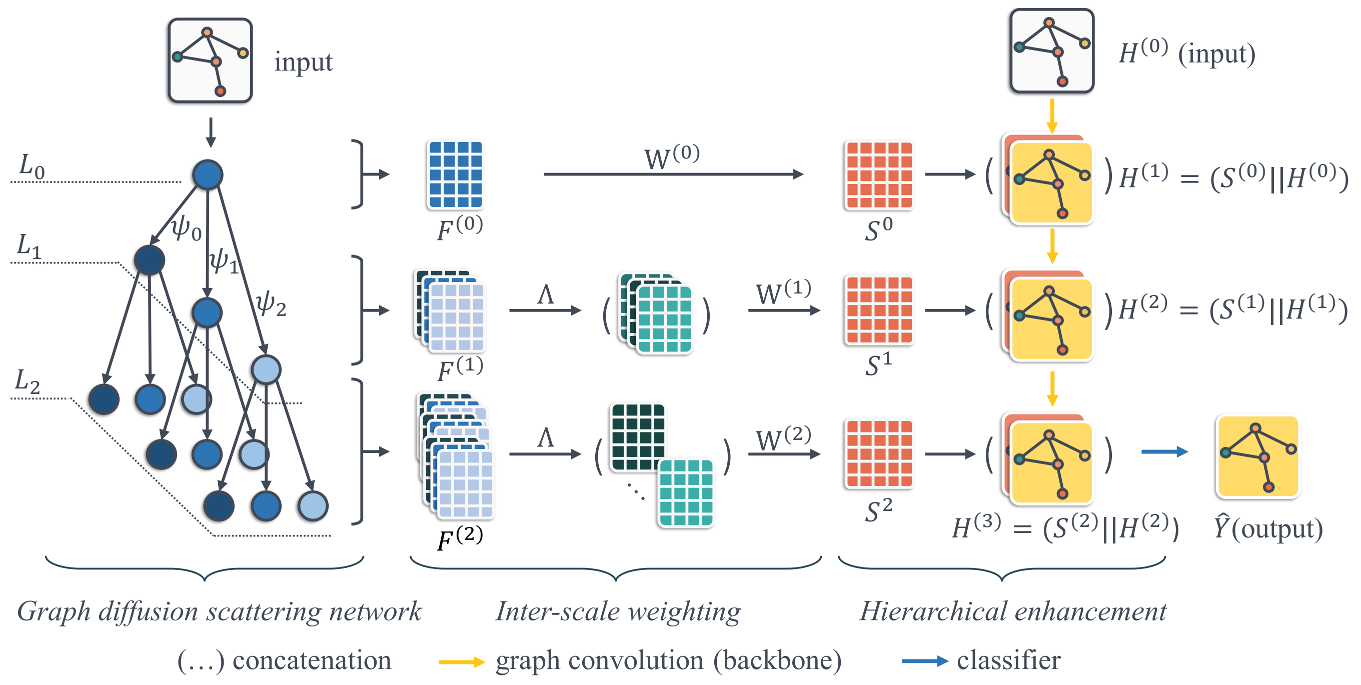

Figure 2: The architecture of proposed HDS-GNN. Illustration for $J = 3$ and $L = 3$. The input graph $\mathcal{G}$ on both sides is the same. The one on the left is the input of the scattering network, and the other one on the right is the input of the backbone GNN. For the first layer L_0, only one scale representation F^0 is obtained so that there is no need for inter-scale weighting. For deeper layers, the shared weight Λ is applied for different scales. At the last layer, we connect a classifier (i.e., a GCN layer) to achieve the node classification task.

to each layer of GNN, and do not consider the progressive relationship between layers, that is, the deeper layer of scattering network can extract more refined features. As shown in Figure 1(c), our method provide finer scattering features for smoother GNN features as the network deepens. This allows the model to maintain sufficiently subtle differences when smooth the overall signal. Our follow-up experiments verifies this observation.

4 The Proposed Method

4.1 The Overall Structure of the Proposed Model

Figure 2 shows the overall structure of the proposed Hierarchical Diffusion Scattering Graph Neural Network (HDS-GNN), where $J = 3, L = 3$ for illustration. The model mainly consists of three parts: 1) *graph diffusion scattering network*, which builds node-level multi-scale rich representation from input graph signal $\mathcal{G}$; 2) *inter-scale weighting*, which makes an adaptive use of different scales with shared weights; and 3) *hierarchical representation enhancement*, which supplements GNN with the scattering features layer by layer to build an enhanced representation. Lastly, a classifier is connected for node classification.

4.2 Node-Level Features of the Scattering Network

In this subsection, we first discuss the properties of the diffusion scattering network (DSN) and then introduce our approach to obtaining node-level scattering features.

Diffusion operator T is constructed by normalized adjacency matrix, and represents diffusion probabilities within

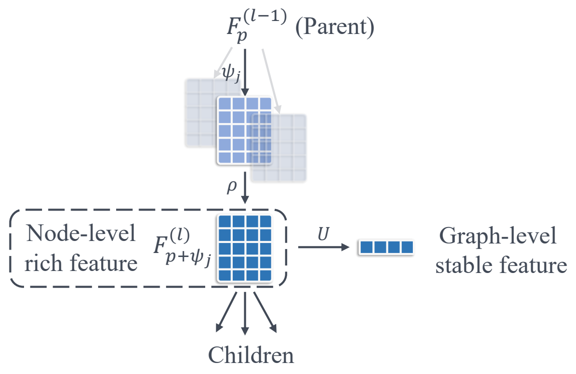

Figure 3: Internal steps of a scattering network layer. The node-level feature (in the dashed box) is extracted for subsequent learning.

one hop (or random walk probability within one step); similarly, T^k represents probabilities within k hops (i.e. k steps). $T^k x$ can be seen as a low-pass filter due to the sum operation on every row of x. As shown in Equ.2, the wavelet operator $\psi_j = T^{2^{j-1}} - T^{2^j}$ represents the difference in probability distribution between radius 2^{j-1} and 2^j, and $\psi_j x$ represents the band-pass signals between the scale 2^{j-1} and 2^j; the base of 2 is for convenient calculation. The signal can be iteratively decomposed by cascading the nonlinearity and wavelet operators to build a multilayer network. The deeper layer can be considered as an enhancement of the previous one, which captures finer information on the previous layer.

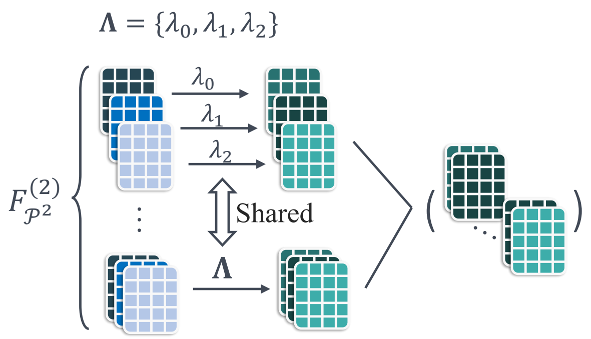
$\mathbf{\Lambda} = \{\lambda_0, \lambda_1, \lambda_2\}$
λ_0
λ_1
λ_2
$F^{(2)}_{\mathcal{P}^2}$
Shared
$\mathbf{\Lambda}$

Figure 4: An illustration of shared inter-scale weighting in one layer. For layer-wise weight sharing, Λ is shared within one layer. For global weight sharing, Λ is also shared with other layers.

Similar to the structure of convolution neural network, which typically is convolution-activation-pooling, diffusion scattering network also consists of three parts: wavelet operators $\mathbf{\Psi}$, nonlinearity ρ, low-pass operator U. Figure 3 shows the detailed steps in one layer of the scattering network. The network extracts the node-level rich features after $\mathbf{\Psi}$ and ρ, and then builds the graph-level stable features after U. To accomplish nodes-level tasks, we apply absolute value as the nonlinearity and take the node-level rich features directly as the scattering features:

$$\begin{cases} F^{(0)}(G,x) = x, \\ F^{(l)}(G,x) = |\mathbf{\Psi} * F^{(l-1)}| = |\mathbf{\Psi} * |\dots|\mathbf{\Psi}x|\dots|| \end{cases} \tag{4}$$

where $|\cdot|$ is point-wise absolute value operator and $F^{(l)}$ is the scattering features in layer l of the scattering network. Each layer of scattering features contains a set of band-pass signal information from scale 1, namely 2^0, to scale 2^J. These multi-scale features correspond to neighbors from near to far in the spatial domain, and frequency bands from high to low in the spectral domain.

4.3 Shared Weighting Across Scales

The previous subsection discusses the scattering network from a horizontal perspective, i.e., the information of every layer and the relationship between layers. From the vertical perspective, the diffusion scattering network is built as a tree structure, so we use "parent" and "child" to represent the relationship between nodes. In this section, we introduce the shared weighting to adaptively regularize different scales.

As shown in Figure 4, we set a learnable weight $\Lambda = \{\lambda_0, \dots, \lambda_J\}$ for every scale to make an adaptive use of them. In this way, the scales containing useful details will be enlarged, while the scales containing noise will be reduced. The main purpose of Λ is to weight different "scales", namely inter-scale weighting. Therefore, Λ is applied to the child nodes from the same parent node and shared across parent nodes in the whole tree, namely *global shared weights*. Considering the progressive relationship between model layers, the weight sharing is also applied within each layer, namely *layer-wise shared weights*. Then, a linear projection is applied on the weighted representation with a learnable matrix $\Theta \in \mathbb{R}^{d \times F_{\text{hid}}}$, where F_{hid} is the dimension of hidden features.

In order to formulate this step, we first define the node in the scattering network. Every non-root node in the scattering network is produced by a sequence of wavelet decomposition, which is a permutation of wavelet operators in the filter bank. We denote this sequence with a scattering path $p = (\psi_{j_1}, \dots, \psi_{j_l})$ where l is the layer number of the node. And we define the one-step path set as $\mathcal{P} = \{(\psi_i); 0 \le i \le J\}$, similarly, the two-step path set is $\mathcal{P}^2 = \{(\psi_i, \psi_j); 0 \le i, j \le J\}$. For completeness, we define the zero-step path set as $\emptyset$, which corresponds to the root of the scattering network. Thus, $\forall p \in \{\emptyset + \mathcal{P} + \mathcal{P}^2 + \dots + \mathcal{P}^l\}$ and we can use p to correspond to every node in the scattering network. Therefore, the inter-scale weighting is defined as:

$$\begin{cases} S^{(0)} = F^{(0)}\Theta^{(0)} \\ S^{(l)} = \underset{p' \in \mathcal{P}^{l-1}}{concat} \left(\left\{ \lambda_k F^{(l)}_{p'+\mathcal{P}[k]} \Theta^{(l)}, 0 \le k \le J \right\} \right) \\ \quad\;\; = \underset{p' \in \mathcal{P}^{l-1}}{concat} \left(\Lambda F^{(l)}_{c(p')} \Theta^{(l)} \right) \end{cases} \tag{5}$$

where *concat* is concatenation operation and $S^{(l)}$ is the weighted DSN feature of layer l; F_p denotes the scattering feature of the node that corresponds to scattering path p; $c(p)$ denotes the child nodes of the node p.

4.4 Hierarchical Representation Enhancement

As we discussed before, the deeper layer of DSN contains the more refined scales and more detailed information, while the deeper layer of GNN contains smoothness over a wider range, which corresponds to a lower frequency signal. Based on this complementary property between DSN and GNN by layers, we propose to enhance GNN representations with DSN features layer by layer. In particular, we concatenate the representations of DSN and GNN layer by layer to build a new representation:

$$H^{(l+1)} = \sigma \left(S^{(l)} \,||\, \text{GNN} \left(A, H^{(l)} \right) \right) \tag{6}$$

where $\sigma(\cdot)$ is a nonlinear activation function and $||$ represents the concatenation operation. For example, if the GCN [Kipf and Welling, 2017] is used as the backbone network (HDS-GCN), the propagation process can be described as:

$$H^{(l+1)} = \sigma \left(S^{(l)} \,||\, D^{-\frac{1}{2}} \widetilde{A} D^{-\frac{1}{2}} H^{(l)} \Theta^{(l)} \right) \tag{7}$$

After the multi-layer feedforward propagation, the node-level embedding fused with multi-scale band-pass signals can be obtained. Then a classifier is followed to accomplish the downstream task. In particular, one GCN layer is connected to achieve semi-supervised node classification:

$$\hat{Y} = \text{softmax} \left(\text{GCN} \left(H^{(L)} \right) \right) \tag{8}$$

$$\mathcal{L} = \text{cross\_entropy} \left(\hat{Y} \left[mask, : \right], Y \left[mask, : \right] \right) \tag{9}$$

As shown in Figure 2, $\hat{Y} \in \mathbb{R}^{n \times C}$ is the output of the classifier, where C is the number of classes; $\mathcal{L}$ is cross entropy loss for node classification; $mask$ is the node mask that only makes training nodes visible.

Datasets	**Cora**	**Citeseer**	**Pubmed**	**DBLP**	**Amazon Computers**	**Amazon Photo**	**Coauthor CS**	**Cornell**	**Texas**
Nodes	2,708	3,327	19,717	17,716	13,381	7,487	18,333	183	183
Edges	5,429	4,732	44,338	105,734	245,778	119,043	81,894	295	309
Features	1,433	3,703	500	1,639	767	745	6,805	1703	1703
Classes	7	6	3	5	10	8	15	5	5
GCN	81.5*	70.3*	79.0*	76.2	83.3	91.7	92.8	52.3	56.9
JKNet	81.1*	69.8*	78.1*	71.2^{+}	83.9^{+}	90.7^{+}	91.3^{+}	54.9^{+}	56.2^{+}
GAT	83.0*	72.5*	79.0*	77.4	81.1	91.3	88.4	56.9	57.4
APPNP	83.3*	71.8*	80.1*	81.1	70.8	90.1	90.2	57.1	56.7
LanczosNet	79.5*	66.2*	78.3*	-	-	-	-	-	-
Sc-GCN	84.2*	71.7*	79.4*	81.5*	85.8	92.9	92.1	60.0	60.9
GSAN	84.0*	71.3*	79.8*	82.6*	81.7	91.1	92.4	62.7	58.2
HDS-GCN$_{(G)}$	**84.6**	72.6	79.9	**84.0**	86.4	92.7	93.5	64.6	67.3
HDS-GCN$_{(L)}$	84.2	**73.0**	79.6	83.7	86.9	94.2	**93.8**	63.7	67.2
HDS-GAT$_{(G)}$	84.3	72.1	**80.6**	82.4	**89.2**	**94.4**	93.5	64.6	68.2
HDS-GAT$_{(L)}$	84.1	71.8	80.3	83.4	89.1	93.7	93.4	**65.5**	**73.6**

Table 1: (1)Datasets statistics (row 1-4). (2)Accuracy results (in percentage) on nine benchmarks (the other rows). The top two results are overlined, and the best results are marked in bold. HDS-*GNN*s are the variants of our model with different backbones. In this experiment, we choose GCN and GAT as the backbones. (G) and (L) represent global shared weights and layer-wise shared weights respectively. Most of the results are from our re-testing with official code; $^{+}$ denotes the results from our re-implementation; * denotes the results from the published papers; OOM denotes Out-of-Memory on our CUDA device.

5 Evalution and Experiment

Datasets

We choose nine benchmarks for experiments: (1) four citation networks: **Cora**, **Citeseer**, **Pubmed** [Sen *et al.*, 2008] and **DBLP** [Bojchevski and Günnemann, 2018]; (2) two co-purchase networks: **Amazon Computers** and **Amazon Photo** [Shchur *et al.*, 2018]; (3) one co-authorship network: **Coauthor CS** [Shchur *et al.*, 2018]. (4) two WebKB networks: **Cornell** and **Texas** [Pei *et al.*, 2019]. The statistical summary can be found in Table 1.

Experimental Settings

Our experimental setup examines the semi-supervised node classification task in the transductive setting. We use sparse splitting (20 per class/500/1000) [Kipf and Welling, 2017] for citation networks, co-purchase networks and co-authorship network, and use dense splitting (60%/20%/20%) [Pei *et al.*, 2019] for WebKB networks. We test all models 10 times and record the average numbers. We use the Adam as the training optimizer and the tool *hyperopt* [Bergstra *et al.*, 2013] for hyper-parameter searching. We set the maximum training epoch to 300 and use early stopping when validation loss does not decrease for consecutive 20 epochs. The learned weights of models used for testing are the checkpoint which has the lowest validation loss in training progress. All the experiments run in PyTorch on NVIDIA 3090.

5.1 Comparison Experiment

We test our model with GCN and GAT as backbones, namely HDS-GCN and HDS-GAT, on the benchmarks mentioned above. Besides, global-shared weighting is denoted by (G) and layer-wise shared weighting is denoted by (L).

Baselines

We choose multiple baseline models on node classification task for a comparison: **GCN** [Kipf and Welling, 2017], **JKNet** [Xu *et al.*, 2018], **GAT** [Veličković *et al.*, 2018], **APPAP** [Klicpera *et al.*, 2018], **LanczosNet** [Liao *et al.*, 2019], **Sc-GCN** [Min *et al.*, 2020] and **GSAN** [Min *et al.*, 2021]. We keeps the experimental results recorded in the published papers as much as possible, and the rest of the results are retested based on the official codes. For JKNet we reimplement a 2$\sim$4 layers model with pyg [Fey and Lenssen, 2019] because no official code was found. The hyperparameter tuning of re-testing models follows the same way with our models and the best results are recorded.

Results

As shown in Table 1, our models achieve the highest accuracy in all used datasets. HDS-GCN achieves the best performance on four datasets (Cora, Citeseer, DBLP, Coauthor CS) and HDS-GAT achieves the best performance on the other five (Pubmed, Amazon Computers, Amazon Photo, Cornell, Texas).

Compared with GCN, our model (HDS-GCN) outperforms by 12.3% at most on Cornell (in absolute accuracy), by 0.9% at least on Pubmed, and by 4.92% on average. Compared with GAT, our model (HDS-GAT) outperforms by 15.8% at most on Texas, by 1.1% at least on Cora, and by 5.87% on average. Our method improves GNN on most datasets, especially on texas and Cornell because these two have more high-frequency information that GNN cannot effectively utilize. It can be noticed that HDS-GAT is outperformed at least 0.4% on Citeseer by GAT. However, in our retesting experiment of GAT with the official code, only an average accuracy

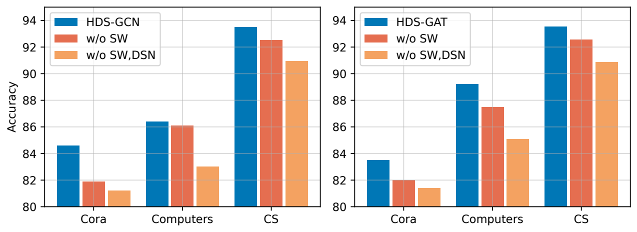

Figure 5: Accuracy results of ablation study on two submodules. Red is for model without *inter-scale weighting* and orange is for model without *diffusion scattering network*.

result of 71.3% is achieved, which is outperformed by HDS-GAT by 0.8% at most.

Compared with Sc-GCN [Min *et al.*, 2020], our model also outperforms it by 6.3% at most on Texas, by 0.4% at least on Cora, and by 2.5% on average. This result verifies that more complete scattering features have better expressive ability.

5.2 Ablation Study

Effect of Submodules

As introduced before, our model can be regarded as three submodules: *diffusion scattering network* (DSN), *inter-scale weighting* (SW) and *hierarchical enhancement*. In this subsection, we start with the complete proposed model and then remove the sub-modules one by one. The submodule *hierarchical enhancement* is actually a fusion approach and not capable for this study and it will be evaluated in the next evaluation. In this experiment, we apply *global shared weights* and set GCN and GAT as the backbones.

We show the results in Figure 5. It is clear that the classification accuracy of the model improves with the addition of every submodule (i.e. from left to right). For most of the datasets, hierarchical scattering features provide more improvement due to the band-pass signals. But for Cora, the scale-weighting provides more improvement, which is because Cora network is assortative [Balcilar *et al.*, 2021b] and the weighting can reduce some unwanted high-frequency information in the added multi-scale signals.

Effect of Fusion Strategies

In this subsection, we aim to make an evaluation of "fusion" and answer a question: does it really work to **hierarchically** fuse **finer features to deeper backbone**? We first evaluate the role of the fusion method and then evaluate the role of the finer features.

(1) Global fusion, or non-hierarchical fusion, provides full-scale scattering representations $\mathcal{S} = concat_{l=0}^{L-1}(S^l)$ to GNN at once. We build two variants to perform comparison. $\mathrm{G_0}$DS-GNN denotes that $\mathcal{S}$ are fused to the first layer of the backbone, i.e. $H^{(0)} = (X \,||\, \mathcal{S})$; $\mathrm{G_L}$DS-GNN denotes that $\mathcal{S}$ are fused to the last layer, i.e. $H^{(L)} = \left(H^{(L-1)} \,||\, \mathcal{S}\right)$. The rest of the experimental settings are the same as the last ablation. As shown in Figure 6, it is obvious that the hierarchical fusion model (blue) outperforms all global fusion models (green). This result demonstrates the effectiveness of hierarchical fusion. Interestingly, in terms of fusing globally, $\mathrm{G_0}$

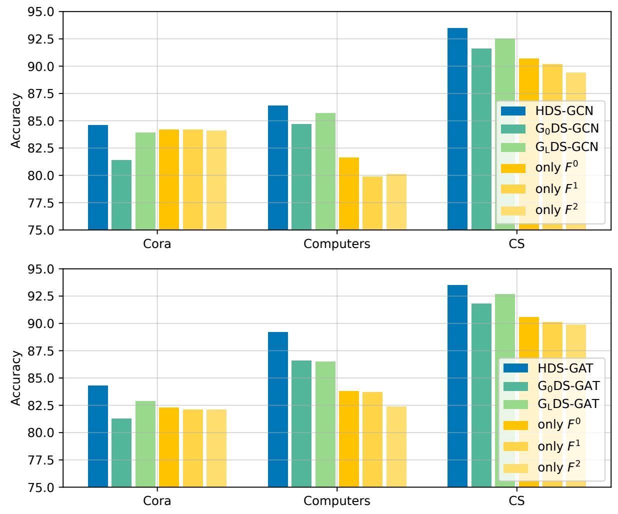

Figure 6: Accuracy results of ablation study on fusion methods. Green is for global fusion and yellow is for scale-invariant fusion.

are outperformed by $\mathrm{G_L}$ in almost all experiments. This is mainly because, after being filtered by GNN, there is only the low frequency of $\mathcal{S}$ left in $\mathrm{G_0}$, which undoubtedly loses the band-pass signal in $\mathcal{S}$.

(2) In order to know if finer scattering features improve deeper GNN layers, we replace all of $F^{(l)}$ with only $F^{(i)}, i < L$ for fusing with scale-invariant band-pass signals. As shown in Figure 6, all the results with scale-invariant signals (yellow) are lower than the results with finer-by-layers signals (blue), which proves our basic observation. Besides, it can be observed the results with finer features are usually lower, which may be because the shallower GNN layers can only obtain coarser latent representations and cannot effectively use finer features. And another interesting observation from the figure is that GDS-GNN performs better than scale-invariant models in most cases, which indicates the full-scale scattering features are usually better than single-scale features (an exception is on the Cora dataset, which because its homophily makes it insensitive to feature scales and more sensitive to fusion method).

6 Conclusion

In this work, we discuss the limitations of current GNN models and elicit an observation of multi-layer GNN and DSN. Based on this, we propose a novel model (HDS-GNN) to augment GNN representation with node-level scattering representations. For future work, we may explore the efficient calculation and more complex fusion method that would also benefit the model.

Acknowledgments

This work is supported by grant BE2019748 Natural Science Foundation of Jiangsu Province and grant 62171125,31800825,31640028 National Natural Science Foundation of China.

References

[Balcilar *et al.*, 2021a] Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *ICML*. PMLR, 2021.

[Balcilar *et al.*, 2021b] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*, 2021.

[Bergstra *et al.*, 2013] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*, pages 115–123. PMLR, 2013.

[Bojchevski and Günnemann, 2018] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[Bruna and Mallat, 2013] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE TPAMI*, 35(8):1872–1886, 2013.

[Coifman and Maggioni, 2006] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and computational harmonic analysis*, 21(1):53–94, 2006.

[Fey and Lenssen, 2019] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[Gama *et al.*, 2018] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Diffusion scattering transforms on graphs. In *ICLR*, 2018.

[Gao *et al.*, 2019] Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *ICML*, pages 2122–2131. PMLR, 2019.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1025–1035, 2017.

[Jiang and Luo, 2021] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *arXiv preprint arXiv:2101.11174*, 2021.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Klicpera *et al.*, 2018] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2018.

[Kong *et al.*, 2021] Youyong Kong, Shuwen Gao, Yingying Yue, Zhenhua Hou, Huazhong Shu, Chunming Xie, Zhijun Zhang, and Yonggui Yuan. Spatio-temporal graph convolutional network for diagnosis and treatment response prediction of major depressive disorder from functional connectivity. *Human brain mapping*, 2021.

[Lee *et al.*, 2019] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, pages 3734–3743. PMLR, 2019.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

[Liao *et al.*, 2019] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. In *ICLR*, 2019.

[Mallat, 2012] Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[Min *et al.*, 2020] Yimeng Min, Frederik Wenkel, and Guy Wolf. Scattering gcn: Overcoming oversmoothness in graph convolutional networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *NIPS*, volume 33, pages 14498–14508. Curran Associates, Inc., 2020.

[Min *et al.*, 2021] Yimeng Min, Frederik Wenkel, and Guy Wolf. Geometric scattering attention networks. In *ICASSP*, pages 8518–8522. IEEE, 2021.

[Nt and Maehara, 2019] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[Pei *et al.*, 2019] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2019.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[Shchur *et al.*, 2018] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *NeurIPS workshop*, 2018.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Xu *et al.*, 2018] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5453–5462. PMLR, 2018.

[Xu *et al.*, 2019] Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. Graph convolutional networks using heat kernel for semi-supervised learning. In *IJCAI*, 2019.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *NIPS*, 31:5165–5175, 2018.

[Zou and Lerman, 2020] Dongmian Zou and Gilad Lerman. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 49(3):1046–1074, 2020.