Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective

Lanning Wei\textsuperscript{1,2,3}, Huan Zhao\textsuperscript{3,*}, Zhiqiang He\textsuperscript{1,4,*}
\textsuperscript{1}Institute of Computing Technology, Chinese Academy of Sciences
\textsuperscript{2}University of Chinese Academy of Sciences \textsuperscript{3}4Paradigm, Inc., \textsuperscript{4}Lenovo Beijing, China

wielanning18z@ict.ac.cn;zhao1@4paradigm.com;hezq@lenovo.com

ABSTRACT
In recent years, Graph Neural Networks (GNNs) have shown superior performance on diverse real-world applications. To improve the model capacity, besides designing aggregation operations, GNN topology design is also very important. In general, there are two mainstream GNN topology design manners. The first one is to stack aggregation operations to obtain the higher-level features but easily got performance drop as the network goes deeper. Secondly, the multiple aggregation operations are utilized in each layer which provides adequate and independent feature extraction stage on local neighbors while are costly to obtain the higher-level information. To enjoy the benefits while alleviating the corresponding deficiencies of these two manners, we learn to design the topology of GNNs in a novel feature fusion perspective which is dubbed F\textsuperscript{2}GNN. To be specific, we provide a feature fusion perspective in designing GNN topology and propose a novel framework to unify the existing topology designs with feature selection and fusion strategies. Then we develop a neural architecture search method on top of the unified framework which contains a set of selection and fusion operations in the search space and an improved differentiable search algorithm. The performance gains on diverse datasets, five homophily and three heterophily ones, demonstrate the effectiveness of F\textsuperscript{2}GNN. We further conduct experiments to show that F\textsuperscript{2}GNN can improve the model capacity while alleviating the deficiencies of existing GNN topology design manners, especially alleviating the over-smoothing problem, by utilizing different levels of features adaptively.\textsuperscript{1}

CCS CONCEPTS
• Information systems → Data mining; • Computing methodologies → Neural networks.

KEYWORDS
Graph Neural Networks, Topology Design, Neural Architecture Search, Over-smoothing, Heterophily

\textsuperscript{1}Lanning is a research intern in 4Paradigm. *: Corresponding author. The implementation of F\textsuperscript{2}GNN is available at: https://github.com/AutoML-Research/F2GNN.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04 . $15.00
https://doi.org/10.1145/3485447.3512185

1 INTRODUCTION
In recent years, Graph Neural Networks (GNNs) have been widely used due to their promising performance in various graph-based applications [16, 34, 44, 49, 56]. In the literature, different GNN models can be built by designing the aggregation operations\textsuperscript{2} and the topology. To improve the model capacity, diverse aggregation operations [19, 23, 41, 49] are designed to aggregate the information from the neighborhood. On the other hand, topology design is also important for the model capacity [10, 25, 50]. One typical topology design manner in GNNs is to stack the aggregation operations. The higher-order neighborhoods can be accessed based on the stacking manner [23], thus, higher-level features can be extracted recursively to increase the model capacity [27]. However, as the network goes deeper, the node representations of connected nodes become indistinguishable, which is called the over-smoothing problem [27]. To address this problem, the identity skip-connection is applied in topology design thus the features of different levels can be utilized to improve the model capacity, e.g., JK-Net [50], ResGCN [25] and DenseGCN [25]. Apart from the stacking manner in GNN topology design, [9, 24] use multiple aggregation operations in each layer to extract features independently and fuse these features to enhance the information extraction of local neighbors. However, multiple aggregations require more resources, hence it is costly to obtain higher-level information when stacking more layers. To summarize, there are two mainstream GNN topology design manners as shown in Figure 1 (i) stacking aggregation operations to obtain the higher-level features (methods on the yellow background); (ii) using multiple aggregation operations to obtain the adequate and independent feature extraction stage on local neighbors. Both of these two manners can improve model capacity. However, the former is easy to get the performance drop due to the over-smoothing problem, and the latter is costly to obtain the higher-level information. Then when designing a GNN model for a specific task, a natural question arises: Can we enjoy the benefits while alleviate the corresponding deficiencies on top of these two topology design manners? In that way, we can further improve the model capacity on top of existing GNN models.

\textsuperscript{2}The aggregation operation in this paper is equivalent to the the message passing layers in GraphGym [51].
However, it is non-trivial to achieve this since there lacks a systematic approach for the GNN topology design in existing works, and it is unknown how to combine the aforementioned two GNN topology design manners. In the literature, the topology of a neural network can be represented by its “computational graph” where the nodes represent the operations and the directed edges link operations in different layers [17, 40, 46, 47, 52]. In this way, the topology of a neural network can be obtained by designing the links among the nodes in the “computational graph”. In existing GNNs, designing the links among operations is equivalent to selecting the features of different levels which proved useful in improving the performance [40, 50, 51]. Nevertheless, the fusion strategy designed to make better utilization of the selected features is also indispensable in improving the GNN model capacity [50, 58, 59]. By reviewing extensive human-designed GNNs, we observe that diverse fusion strategies are employed to integrate the features generated by linked operations. As the representative GNNs shown in Figure 1, ResGCN [25] adds up the residual features in each layer, JK-Net [50] provides three fusion operations (maximum for example) to integrate the representations of the intermediate layers, and PNA [9] concatenates the results of multiple aggregation operations. Therefore, in designing the topology of GNNs, it is important to take the links as well as the fusion strategies into consideration.

On top of the existing GNNs, we design a novel framework to unify the GNN topology designs with feature selection and fusion strategies. Therefore, the topology design target is transformed into the design of these 2 strategies. As shown in Figure 2, without loss of generality, the framework is represented as a directed acyclic graph (DAG), which is constructed with an ordered sequence of blocks. Based on this framework, diverse topology designs including existing works can be modeled by different selection and fusion strategies.

Then, another challenge is to design adaptive feature selection and fusion strategies to achieve the SOTA performance given a specific graph-based task, since the preferable GNN topology can significantly differ across different datasets [51]. Therefore, to address this challenge, we borrow the power of neural architecture search (NAS), which has been successful in designing data-specific CNNs [30, 61] and GNNs [15, 59], to achieve the adaptive topology design. To be specific, we firstly propose a novel search space that contains a set of selection and fusion operations, and then develop an improved differentiable search algorithm based on a popular one, i.e., DARTS [30], by addressing the obvious optimization gap induced by two opposite operations in the search space. Finally, we extract the optimal strategies based on the unified framework when the searching process terminates, then an adaptive topology design is obtained.

In this paper, we learn to design the topology of GNNs in a novel feature fusion perspective and it can be dubbed F^2GNN (Feature Fusion GNN). Extensive experiments are conducted by integrating the proposed method with predefined and learnable aggregation operations on eight real-world datasets (five homophily and three heterophily), then the performance gains demonstrate the effectiveness of the proposed method. Furthermore, we conduct experiments to evaluate the advantages of F^2GNN in designing the topology of GNNs, from which we can observe that F^2GNN can enjoy the benefits and alleviate the deficiencies of existing topology design manners, especially alleviating the over-smoothing problem, by utilizing different levels of features adaptively.

To summarize, the contributions of this work are as follows:

- In this paper, we provide a novel feature fusion perspective in designing the GNN topology and propose a novel framework to unify the existing topology designs with feature selection and fusion strategies. It transforms the GNN topology design into the design of this two strategies.
- To obtain the adaptive topology design, we develop a NAS method on top of the unified framework containing a novel search space and an improved differentiable search algorithm.
- Extensive experiments on eight real-world datasets (five homophily and three heterophily) demonstrate that the proposed F^2GNN can improve model capacity (performance) while alleviating the deficiencies, especially alleviating the over-smoothing problem.

Notations. We represent a graph as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the node and edge sets. $A \in \mathbb{R}^{\mid\mathcal{V}\mid \times \mid\mathcal{V}\mid}$ is the adjacency matrix of this graph where $\mid\mathcal{V}\mid$ is the node number. The class of node $u$ is represented as $y_u$. In the proposed framework, the output features of block $i$ are denoted as $H^i$, and in other methods, $H^i$ represents the output features of layer $i$. $H^{output}$ represents the output of a GNN.

2 RELATED WORK

2.1 Topology Designs in Graph Neural Network

GNNs are built by designing the aggregation operations and topologies. One mainstream topology design manner is to stack aggregation operations [19, 23, 41]. The high-level information can be captured to improve the model capacity while easily resulting in the over-smoothing problem in deeper networks [27]. To address this problem and improve the model capacity, the identity skip-connections are provided additionally to integrate different levels of features. Jk-Net [50] and DAGNN [31] integrate the features of all the intermediate layers at the end of GNNs; ResGCN [25] and DenseGCN [25] have the same connection schemes as ResNet and DenseNet; GCNII [6] adds up the initial features in each layer. Apart from the stacking manner, multiple aggregation operations provide adequate and independent feature extraction on the local neighbors to improve the model capacity. PNA [9] and HAG [24] provide multiple aggregation operations in each layer to learn features from local neighbors independently; MixHop [1], IncepGCN [36] and...
InceptionGCN [22], which are the inception-like methods, provide multiple aggregations in each branch to extract different levels of features independently. However, multiple aggregations require more resources, hence it is costly to obtain higher-level information.

Considering the benefits and deficiencies of the existing topology designs, we propose a novel method $F^2$GNN in the feature fusion perspective to design the topology of GNNs adaptively. It can unify the existing topology designs and provide a platform to explore more topology designs, which are more expressive than these human-designed ones.

### 2.2 Graph Neural Architecture Search

NAS methods use a search algorithm to find the SOTA neural networks automatically in a pre-defined search space and representative methods are [26, 30, 48, 61]. Very recently, researchers tried to automatically design GNNs by NAS. These methods mainly focus on designing the aggregation operations on top of the vanilla stacking manner, e.g., GraphNAS [15] provides the attention function, attention head number, embedding size, etc. Similar search spaces are also used in [29]. Several methods further provide the skip-connection design based on this stacking manner, e.g., SNAE [58] and SNAG [59] are built on JK-Net [50] and learn to select and fuse the features of intermediate layers in the output node. AutoGraph [28] is built on DenseNet [21] and learns to select features in each layer. GraphGym [51] provides the residual feature selection and fusion strategies in designing GNNs on top of the stacking manner. Besides these methods, RAN [40] learns to design the GNN topologies. It uses the computational graph to represent the GNN topology and then designs the links with a randomly-wired graph generator [47]. However, it lacks the explorations of fusion strategies that can improve feature utilization in GNNs.

Apart from the search space designs, diverse search algorithms are provided to search the SOTA neural networks from the proposed search space, e.g., incorporating the Reinforcement Learning (RL) into the searching strategy [15, 58, 61] or using the Evolutionary Algorithm (EA) directly [5, 18, 28]. Considering the search efficiency, the differentiable algorithm is proposed to search architectures with gradient descent. It relaxes the discrete search space into continuous and then treats the architecture search problem as a bi-level optimization problem [3, 29, 30, 59].

More graph neural architecture search methods can be found in [11, 35, 42–44, 54–57]. Compared with existing methods which mainly focus on designing aggregation operations, in this work we explore the additional topology design, which can thus be regarded as an orthogonal and complementary approach to improve the GNN model capacity.

### 3 FEATURE FUSION FRAMEWORK

In this section, we elaborate on the proposed feature fusion framework and show how to translate this framework into diverse GNNs.

#### 3.1 The Proposed Unified Framework

In the literature, GNNs can be built by designing the aggregation operation and its topology. The topology is designed in two major manners: stack aggregation operations to obtain higher-level features or use multiple operations to provide rich and independent local feature extractions. By reviewing existing human-designed GNNs, we observe that features of different levels are selected due to diverse selection strategies, and it leads to the utilization of diverse fusion strategies to integrate these features [1, 9, 25, 50]. In other words, the feature selection and fusion strategies lead to the key difference of topology designs in GNNs. Therefore, the topology designs utilized in existing methods can be unified with these two strategies.

Based on this motivation, we propose a feature fusion framework to unify these two topology design manners. As shown in Figure 2, without loss of generality, the framework is represented as a DAG which is constructed with an ordered sequence of one input block, $N$ SFA blocks ($N = 4$ for example; SFA: selection, fusion and aggregation), and one output block. The input block only contains a simple pre-process operation, i.e., Multilayer Perceptron (MLP) in this paper, that supports the subsequent blocks. The SFA block contains the selection, fusion and aggregation operations. For the $i$-th SFA block as shown in Figure 2(b), there exists $i$ predecessors thus $i$ feature selection operations $f_i$ are utilized to select the features generated by the previous blocks. One fusion operation $f_f$ is used to fuse these selected features, and one aggregation operation $f_a$ is followed to aggregate messages from the neighborhood. Therefore, the high-level features can be generated by $H^i = f_a(f_f(f_i(H^0), \cdots, f_i^{t-1}(H^{t-1})))$. In the output block, a 2-layer MLP that serves as the post-process operation is provided after the $N + 1$ selection operations and one fusion operation. On top of the unified framework, the topology design is transformed into the design of selection and fusion strategies. Compared with existing methods that focus on designing aggregation operations, our framework provides a platform to explore the GNN topology designs which is more expressive than existing methods.

Figure 2: (a) The proposed framework consists of an ordered sequence of one input block, $N$ SFA blocks (four as an example here) and one output block. (b) For the $i$-th SFA block, we have $i$ selection operations $f_i$ and one fusion operation $f_f$ to utilize the features generated by $i$ predecessors. Then one aggregation operation $f_a$ is followed to aggregated messages from the neighborhood.
we develop a new algorithm to alleviate this problem. Which contains the selection and fusion operation sets is provided. As the commonly-recognized paradigm of NAS, the search space

3.2 Translating the framework into diverse GNNs

The feature fusion framework can unify the existing topology designs thus we can translate the framework into diverse GNNs. The formulation and illustrations of the most closely related methods are shown in Figure 3. For simplicity, the feature transformation is ignored. Vanilla GNNs [19, 23, 41, 49] are constructed by stacking aggregation operations and these methods can be approximated as the single path in our framework. ResGCN [25] and JK-Net [50] provide extra identity skip-connections to utilize the different levels of features, thus diverse selection and fusion operations are utilized as illustrated in Figure 3 (b) and (c), respectively. GCNII [6] uses the initial residual connections to address the over-smoothing problem, and it can be approximated as in Figure 3 (d). PNA [9] uses M aggregation operations in each layer, and it can be approximated by M branches in this framework. MixHop [1] concatenates the features based on the power of adjacency matrix \( A \). For the graph neural architecture search methods which focus on design aggregation operations [15, 29] or provide additional skip-connection learning [58, 59] on top of the stacking manner, they can be treated as the variants of vanilla and JK-Net, respectively. Thus, we can see the advantage of the proposed framework, which can provide a systematic and unified view of existing GNN topology designs. Next, we show how to obtain the data-specific GNN topology design on top of this unified framework.

4 DESIGN GNNS WITH THE FRAMEWORK

On top of the unified framework, generating one adaptive topology design is a non-trivial task since it is still unknown how to design the optimal feature selection and fusion strategies given a specific task. In this paper, we adopt the NAS method to address this challenge. As the commonly-recognized paradigm of NAS, the search space which contains the selection and fusion operation sets is provided. Then the topology design can be decomposed into the decision of operations in the search space. To demonstrate the versatility of our method, we further provide the aggregation operation set thus we can design GNNs with topologies and aggregation operations simultaneously. The differentiable search algorithm [30] is widely used considering the efficiency and effectiveness. Nevertheless, we observe the performance gap in applying this algorithm due to the two opposite selection operations, i.e., ZERO and IDENTITY. Thus, we develop a new algorithm to alleviate this problem.

### Table 1: The formulation and illustrations of the closely related methods.

| Method      | Equation                                      |
|-------------|-----------------------------------------------|
| Vanilla GNN | \( H^{m+1} = f_a(H^m) \)                     |
| ResGCN [25] | \( H^{m+1} = f_a(H^m + H^{m-1}) \)            |
| JK-Net [50] | \( H^{m+/pred} = f_z(H^0; \cdots; H^m) \)     |
| GCNII [6]   | \( H^{m+1} = \alpha H^m + (1 - \alpha) f_a(H^m) \) |
| PNA [9]     | \( H^{m+1} = \sum_{e \in M} A_e H^m \)       |
| MixHop [1]  | \( H^{m+1} = \sum_{e \in M} A_e H^m \)       |

Figure 3: The formulation and illustrations of the closely related methods. \( \odot \) denotes the summation operation and \( \otimes \) denotes the concatenation operation.

4.1 Search Space

One GNN can be built by designing the aggregation operations and the topology. The aggregation operations are widely explored in existing methods. In general, GNNs use the same aggregation operation in each layer. [15, 58, 59] learn to design the layer-wise aggregations with the help of the NAS methods as introduced in Section 2.2. Therefore, in this paper, we design the topology designs on these two aggregation variants.

4.1.1 Predefined aggregation operation. The predefined aggregation is utilized in each SFA block and then we need to design the selection and fusion operations in the framework with the help of NAS. After the search terminates, we can obtain the designed GNN with the given aggregation operation. We provide 2 aggregation operations GraphSAGE and GAT thus this method can be dubbed \( F^2 \) SAGE and \( F^2 \) GAT, respectively. Other aggregation operations like GCN and GIN can be trivially integrated with the proposed framework. Without loss of generality, we provide a set of candidate selection operations \( O_s \) and fusion operations \( O_f \) in the search space as shown in Table 1.

#### Selection Operations. For the selection operation, there are only “selected” and “not selected” stages for each feature in existing methods. Thus, we provide two operations IDENTITY and ZERO to select features, which can be represented as \( f(h) = h \) and \( f(h) = 0 \cdot h \), respectively.

#### Fusion Operations. In SFA block and the output block, one fusion operation is needed to fuse the selected features. Based on the literature, we provide six fusion operations to fuse these features with the summation, average, maximum, concatenation, LSTM cell and attention mechanism, which are denoted as SUM, MEAN, MAX, CONCAT, LSTM and ATT, respectively.

4.1.2 Learnable aggregation operation. Compared with the existing graph neural architecture search methods which focus on designing the aggregation operations, we provide the extra aggregation operation set \( O_a \) thus we can design GNNs with topologies and aggregation operations simultaneously. This method is dubbed \( F^2 \) GNN.

#### Aggregation Operations. Four widely used aggregation operations are used in this paper: GCN [23], GAT [41], GraphSAGE [19] and GIN [49], which denoted as GCN, GAT, SAGE and GIN, respectively. In this paper, we focus on designing the topology of GNNs thus only four aggregations are provided. More operations in existing methods can be trivially added if needed.
4.2 The Improved Search Algorithm

Based on the proposed framework and the search space, the search algorithm is used to search operations from the corresponding operation set. Considering the efficiency and effectiveness which have been demonstrated in [30, 59], without loss of generality, the differentiable search algorithm is employed.

Preliminary: Differentiable Architecture Search. A supernet is defined to subsume all models on top of the unified framework and the search space, and it is achieved by mixing the candidate operations [18, 30]. The results of the mixed operation can be calculated by a weighted summation of all candidate operations which denoted as \( \hat{E}(x) = \sum_{k=1}^{O} c_k \cdot \hat{E}_k(x) \), where \( c_k \in (0, 1) \) is the weight of \( k \)-th candidate operation \( \hat{E}_k(x) \). In general, the operation weight \( c_k \) is generated by one relaxation function \( c_k = \exp(\alpha_k) / \sum_{\nu=1}^{O} \exp(\alpha_{\nu}) \) and \( \alpha_k \) is the corresponding learnable supernet parameter for \( c_k \) [30]. Based on the relaxation function, the discrete selection of operations is relaxed into continuous and we can generate the final results in the output block step by step as shown in Figure 2. Thus the supernet can be optimized with gradient descent which can accelerate the search process in orders of magnitude. After finishing the search process, we preserve the operations with the largest weights in each mixed operation, from which we obtain the searched GNN.

The optimization gap in feature fusion. We optimize the supernet in the search process and then derive the GNN after the search is finished. However, it is difficult to generate the best childnet from the supernet since we optimize the supernet in the search process and only select the childnet in reality, which is called optimization gap [8, 45] and the evaluation results can be found in Section 5.4.2. The performance drop caused by the optimization gap is extremely obvious in the our method since we provide two opposite operations, and we cannot derive the best childnet from the supernet due to this gap as the evaluation in Section 5.4.2.

Improved search with the usage of temperature. Considering the influence of IDENT operation, we add a temperature in Softmax function as \( c_k = \exp(\alpha_k / \lambda) / \sum_{\nu=1}^{O} \exp(\alpha_{\nu} / \lambda) \). Thus, with a small temperature \( \lambda \), the operation weight vector \( e \) close to a one-hot vector, and the results of mixed selection operation in Eq. (1) close to a zero tensor when the \( \text{ZERO} \) operation is selected. That is, the IDENT operation will have a smaller influence on the selection results when the \( \text{ZERO} \) operation is chosen, and the optimization gap in our method can be alleviated. Similar solutions can be found in [7, 48]. In this paper, we set \( \lambda = 0.001 \) and the influence of different temperatures will be shown in Section 5.4.2.

Deriving process. The architecture searching task is treated as the bi-level optimization problem. Our method is optimized with gradient descent introduced in [30, 59]. More details can be found in Appendix A. After finishing the search process, we preserve the operation with the largest weight in each mixed operation, from which we obtain the searched architecture.

5 EXPERIMENTS

5.1 Experimental Settings

Datasets. Existing GNNs assume strong homophily where neighbors are in the same class. The homophily ratio can be calculated by \( h = \frac{|\{(u, v) \in E | u \in C \land v \in C \}|}{|E|} \) which is the fraction of edges in a graph which connect nodes that have the same class label. As shown in Table 2, five widely used homophily datasets (higher homophily ratio \( h \)) and three heterophily datasets (lower homophily ratio \( h \)) are selected to evaluate the performance of our method. More introductions about these datasets can be found in Appendix B.1. The comparisons between the F^2GNN and other heterophily methods [12, 60] will be considered in future work.
Baselines. On top of the predefined aggregations, we provide nine GNNs constructed with different topologies. (a) GNNs are constructed by stacking two and four aggregation operations; (b) based on the stacking manner, we construct 4-layers GNNs on top of three commonly used topology designs ResGCN [25], DenseGCN [27] and JK-Net [50]. They are denoted as RES, DENSE and JK, respectively; (c) 4-layer GNNII is constructed based on the topology shown in Figure 3(d); (d) the topology of 1-layer PNA and MixHop are shown in Figure 2(e) and Figure 2(f), respectively. The PNA and MixHop baselines in our experiment are constructed by stacking two layers. (e) the topology designs are constructed by selecting operations from the search space randomly, which is denoted as Random in our experiments.

Compared with F$^2$GNN which designs the topology and aggregation operations simultaneously, we provide three graph neural architecture search baselines: (a) an RL based method SNAG [58], (b) a differentiable method SANE [59], and (c) a random search algorithm that uses the same search space as F$^2$GNN.

More details of these baselines are provided in Appendix B.2.

Implementation details. For Cora, DBLP, Computers, PubMed and Physics, we split the dataset with 60% for training, 20% for validation and test each considering supernet training and evaluation. For Wisconsin and Actor, we adopt the 10 random splits used in [34, 60](48%/32%/20% of nodes per class for train/validation/test). For Flickr dataset, we adopt the split in [53](50%/25%/25% of nodes per class for train/validation/test). For all NAS methods (Random baselines, SNAG, SANE and our method), we search a GNN with the corresponding search space. Then all the searched GNNs and the human-designed baselines are tuned individually with hyperparameters like embedding size, learning rate, dropout, etc. With the searched hyperparameters, we report the average test accuracy and the standard deviation on 10 repeated results. More details about the implementation stage are shown in Appendix B.3.

5.2 Performance Comparisons

The results are given in Table 3. Firstly, based on two predefined aggregation operations, there is no absolute winner among eight human-designed GNNs constructed with existing topology designs. Among them, with the utilization of different feature selection and fusion strategies, six baselines have a better performance than two stacking baselines in two groups in general. The performance gain demonstrates the importance of feature utilization in improving the model capacity. Secondly, by designing GNN topologies with adaptive feature selection and fusion strategies, the proposed method can achieve the top rank on two predefined aggregations. In particular, the SOTA performance is achieved on seven out of eight datasets. Thirdly, with the same search space, the Random baselines also achieve considerable performance gains on all these datasets, which demonstrates the effectiveness of the unified framework in Section 3. Nevertheless, the Random baseline is outperformed by the proposed method, which indicates the usefulness of the improved search algorithm in designing the topology of GNNs.

On the other hand, looking at the results of the learnable aggregation operations, SANE and SNAG only focus on selecting features in the output block based on a stacking manner (A sketch of the topology is given in Figure 8 in Appendix B.2.). These methods have limited topology design space and can be treated as special instance of F$^2$GNN. Compared with these two methods, F$^2$GNN and Random methods design GNNs on the proposed unified framework and achieve higher performance. Thus it demonstrates the effectiveness of the adaptive topology design, i.e., feature selection and fusion strategies based on the unified framework. Besides, [30, 59] have shown the efficiency advantage of the differentiable methods (our method and SANE) over those RL and randomly based methods. We also obtained the two orders of magnitude less search cost in our experiments (The details of the search cost comparison are shown in Section 5.4.2.). These 2 differentiable methods stay at the top of the rank list (F$^2$GNN ranks 2.75 and SANE ranks 6.38), which indicates the power of differentiable search algorithms in designing GNNs.

Searched topologies. We visualize the searched topologies on Cora, Physics and Actor datasets with two human-designed aggregation operations: GraphSAGE and GAT. The index of each block in the framework is annotated and the dark blocks indicate they are not used in the searched GNNs.

Figure 4: The searched topologies on Cora, Physics and Actor datasets with two human-designed aggregation operations: GraphSAGE and GAT. The index of each block in the framework is annotated and the dark blocks indicate they are not used in the searched GNNs.
Table 3: Performance comparisons of our method and all baselines. We report the average test accuracy and the standard deviation with 10 splits. “L2” and “L4” mean the number of layers of the base GNN architecture, respectively. The best result in each group is highlighted in gray, and the second best one is underlined. The group accuracy rank and the overall accuracy rank of each method are calculated on each dataset. The average rank on all datasets is provided. The Top-2 methods in each group and the Top-3 methods are calculated on each dataset. The average rank on all datasets is provided. The Top-2 methods in each group and the Top-3 methods in this table are highlighted in gray.

| Aggregation | Topology | Cora | DBLP | PubMed | Computers | Physics | Actor | Wisconsin | Flickr | L4 | L6 | L8 | L10 |
|-------------|----------|------|------|--------|-----------|--------|-------|-----------|--------|-----|----|----|-----|
| SAGE        | Stacking (L2) | 86.09(0.50) | 83.80(0.33) | 88.96(0.29) | 91.14(0.30) | 96.42(0.11) | 34.78(1.10) | 79.61(5.36) | 51.21(0.71) | 6.63 | 15.00 |
|             | Stacking (L4) | 85.60(0.61) | 83.83(0.32) | 88.23(0.28) | 90.52(0.42) | 95.97(0.14) | 33.61(1.08) | 60.39(10.77) | 53.07(0.50) | 8.25 | 17.00 |
|             | RES (L4) | 85.60(0.52) | 83.90(0.30) | 88.99(0.25) | 91.51(0.18) | 96.31(0.17) | 35.16(0.94) | 76.47(5.26) | 53.72(0.27) | 5.25 | 13.13 |
|             | DENSE (L4) | 86.80(0.59) | 83.30(0.70) | 89.42(0.27) | 90.74(0.53) | 96.38(0.14) | 34.78(0.60) | 77.06(6.01) | 53.17(0.19) | 4.50 | 12.75 |
|             | MixHop (L2) | 84.70(0.70) | 83.94(0.62) | 89.21(0.29) | 91.21(0.30) | 96.56(0.05) | 36.53(0.92) | 81.96(4.71) | 52.45(0.31) | 4.75 | 10.38 |
|             | GNNII (L4) | 85.30(0.42) | 84.48(0.45) | 89.21(0.24) | 91.38(0.27) | 96.45(0.15) | 35.70(1.11) | 81.57(4.13) | 52.24(0.29) | 4.50 | 11.50 |
|             | MixHop (L2) | 84.29(0.67) | 82.76(0.72) | 89.25(0.26) | 90.67(0.42) | 96.32(0.10) | 33.89(2.68) | 75.29(6.46) | 52.09(0.73) | 8.88 | 17.75 |
|             | PNA (L2) | 86.81(0.95) | 82.65(0.65) | 89.25(0.28) | 88.56(1.61) | 96.11(0.17) | 36.30(0.58) | 85.10(5.63) | 54.10(0.15) | 3.50 | 8.75 |
|             | MixHop (L2) | 86.75(0.29) | 83.60(0.29) | 89.21(0.04) | 91.30(0.19) | 96.46(0.03) | 36.61(1.00) | 85.88(1.72) | 53.66(0.16) | 2.00 | 4.38 |
|             | Random | 86.72(0.26) | 84.81(0.06) | 89.73(0.26) | 91.80(0.26) | 96.72(0.04) | 36.61(1.00) | 85.88(1.72) | 53.66(0.16) | 2.00 | 4.38 |

- We can benefit from the multiple aggregation design manner which provides adequate and independent local feature extractions, e.g., two aggregation operations are selected in the second layer on Cora and the first layer on Physics.
- On Actor, the representative heterophily dataset, we obtained an MLP network based on F^2SAGE. This topology design is consistent with H^2GCN [60] which shows that the graph structure is not always useful for the final performance, and it further demonstrates the effectiveness and the versatility of our method. More searched topologies are shown in Appendix B.4 and more results about heterophily datasets are given in Appendix B.5.

5.3 Advantages of the Adaptive Topology Design

In designing the topology of GNNs, stacking aggregation operations devoted to obtaining higher-level features but easily got the over-smoothing problem as the network goes deeper, and the multiple aggregation operations provide adequate and independent feature extraction stage on local neighbors while are costly to obtain the higher-level information. The performance gains in Table 3 and the effectiveness of our method is demonstrated further. It further demonstrates the effectiveness of our method.

Alleviating the over-smoothing problem. As the network goes deeper, the node representations become indistinguishable and easily obtained higher-level information. The performance gains in Table 3 and the effectiveness of our method is demonstrated further. It further demonstrates the effectiveness of our method.

In designing the topology of GNNs, stacking aggregation operations devoted to obtaining higher-level features but easily got the over-smoothing problem as the network goes deeper, and the multiple aggregation operations provide adequate and independent feature extraction stage on local neighbors while are costly to obtain the higher-level information. The performance gains in Table 3 and the effectiveness of our method is demonstrated further. It further demonstrates the effectiveness of our method.

Alleviating the over-smoothing problem. As the network goes deeper, the node representations become indistinguishable and easily got performance drop, which is called over-smoothing [27]. MAD (Metric for Smoothness) [4] is used to measure the smoothness of the features. In Figure 5, we show the comparisons of test accuracy and MAD value on the Cora dataset on the conditions of different layers and SFA blocks. For comparisons, on top of the GraphSAGE, we provide three topologies that have been proved helpful in alleviating the over-smoothing problem, and more results can be found in Appendix B.6. Other methods [4, 13, 36] which can alleviate this problem will be left into future work. In Figure 5, RES, JK and MixHop can achieve stable performance and higher MAD values compared with the stacking baseline. It demonstrates the effectiveness of different layers of features in alleviating the over-smoothing problem. Compared with these baselines, F^2SAGE can achieve the best performance and higher MAD values by utilizing features in each block adaptively. It can further indicate the effectiveness of our method.

Flexibility in obtaining the higher-level features. The multiple aggregation operations provide an adequate and independent feature
In this section, we conduct ablation studies on the proposed framework.

### 5.4 Ablation study

In this section, we conduct ablation studies on the proposed framework.

5.4.1 The Importance of Fusion Strategy. In designing topologies, existing methods mainly focus on the links among operations [40, 47]. By reviewing the extensive human-designed GNNs, we observe that the feature selection and fusion strategies result in the main difference of topology designs in GNNs. It indicates that the selection and fusion strategies are key components in designing the topology of GNNs, and the fusion strategies should be considered since they can improve the utilization of features in different levels [50, 58, 59]. To evaluate the importance of fusion strategy, we learn to design topologies with fixed fusion operations instead. As shown in Table 4, with the three predefined fusion operations, the performance drop is evident on three variants, which demonstrates the importance of the fusion strategy. Therefore, designing the topology of GNNs with the selection and fusion operations is significant compared with the existing methods.

5.4.2 The Evaluation of the Improved Search Algorithm. The efficiency of the differentiable search algorithm over others is significant and has been proved in existing methods [29, 30, 59]. Therefore, we provide an improved differentiable search algorithm to design GNNs. In Table 3, SNAG and Random methods need to sample architectures and then train from scratch. In our experiments, they require 26.44 and 0.903 GPU hours to search GNNs on the Cora dataset. SANE and $F^2$GNN employ the differentiable search algorithm and the search cost are 0.007 and 0.028 GPU hours on the Cora dataset, respectively.

However, we observe that the optimization gap has a large influence on the feature selection operation due to the two opposite selection operations. As shown in Table 5, we use the validation accuracy to quantify the optimization gap in the feature fusion framework. These accuracies are obtained at the end of the search and after architecture derivation without fine-tuning. We can observe that when the temperature is too large, i.e., 1 and 0.1, the validation accuracy gap of the supernet and the childnet is large when we search the feature selection and fusion operations. In these cases, the ZERO operation is selected for most features. However, in the $F^2$AGG method, which only searches aggregation operations based on the selection operation IDENT and fusion operation SUM, the accuracy gap is much smaller. Therefore, considering the two opposite operations in the selection operation set, we use the small temperature of 0.001 instead. The performance gains in Table 3 indicate the effectiveness of the improved search algorithm in addressing the optimization gap in feature fusion and obtaining the expressive GNNs.

5.4.3 The Influence of the Number of SFA Blocks. In this paper, we only use four SFA blocks for example. Here we conduct experiments to show the influence of the number of SFA blocks.
Table 5: Comparisons of the validation accuracy in the supernet and the childnet on the Cora dataset.

| Temperature | F^2SAGE Supernet | F^2SAGE Childnet | F^2AGG Supernet | F^2AGG Childnet |
|-------------|------------------|------------------|----------------|----------------|
| 1           | 80.33            | 86.83            | 80.15          | 86.83          |
| 0.1         | 73.65            | 84.23            | 84.60          | 84.60          |
| 0.01        | 70.13            | 84.60            | 84.60          | 84.60          |
| 0.001       | 80.15            | 86.83            | 86.83          | 86.83          |

Figure 7: Performance comparisons of the different SFA block numbers on $F^2SAGE$. Different colors represent different block numbers.

In Figure 7, $F^2SAGE$ achieves a stable performance on the condition of different SFA blocks, yet the stacked baselines obtained the performance drop due to the over-smoothing problem as shown in Figure 5. The increasing number of SFA blocks do not bring about the performance drop due to the adaptive utilization of different levels of features, which then demonstrates the strong ability of our method in alleviating the over-smoothing problem.

6 CONCLUSION AND FUTURE WORK

In this paper, we provide a novel feature fusion perspective in designing the GNN topology and propose a novel framework to unify the existing topology designs with feature selection and fusion strategies. In this way, designing the topology of GNNs is transformed into designing the feature selection and fusion strategies in the unified framework. To obtain an adaptive topology design, we develop a NAS method. To be specific, we provide a set of candidate selection and fusion operations in the search space and develop an improved differentiable search algorithm to address the obvious optimization gap in our method. Extensive experiments are conducted on the predefined and learnable aggregation operations on eight real-world datasets (five homophily and three heterophily). The results demonstrate the effectiveness and versatility of the proposed method, and we can enjoy the benefits and alleviate the deficiencies of existing topology design manners, especially alleviating the over-smoothing problem, by utilizing different levels of features adaptively.

For future work, we will investigate the influence of different candidate operations and algorithms, and explore $F^2GNN$ in large-scale graphs [20] and the heterophily datasets.

ACKNOWLEDGMENTS

We thank Dr. Quanming Yao and Dr. Yongqi Zhang to improve the method and manuscript. We also thank all anonymous reviewers for their constructive comments, which help us to improve the quality of this manuscript.
Algorithm 1 F^2GNN - Feature Fusion GNN

Require: Input graph $\mathcal{G} = (V, E)$, the adjacency matrix $\mathbf{A}$ and the feature matrix $\mathbf{H}^{input}$. Training set $\mathcal{D}_{train}$ and validation set $\mathcal{D}_{valid}$, the epoch $T$ for search.

Ensure: The searched architecture.

1: Random initialize the parameters $\alpha$ and $\mathbf{W}$.
2: for $t = 1$ to $T$ do
3: Calculate the operation weights $\mathbf{C}$ on top of $\alpha$.
4: $\mathbf{H}^0 = \text{MLP}(\mathbf{H}^{input})$
5: for $i = 1$ to $N + 1$ do
6: $\mathbf{H}^i = \{\}$
7: for $j = 0$ to $i - 1$ do
8: $f^i_j(\mathbf{H}) = \sum_{k=1}^{N} c^i_k o^j_k (\mathbf{H}) = c^i_j \mathbf{H}^{j//Selection}$
9: $\mathbf{H}^i = \mathbf{H}^i \cup \{f^i_j(\mathbf{H})\}$
10: $\mathbf{H}^i = f^i_j(\mathbf{H}^i) = \sum_{i=1}^{N} c^i_k o^j_k (\mathbf{H}^i) //\text{Fusion}$
11: if $i = N + 1$ then
12: $\mathbf{H}^{output} = \text{MLP}(\mathbf{H}^i)$
else
13: $\mathbf{H}^i = f^i_j(\mathbf{H}^i, \mathbf{A}) //\text{Aggregation}$
15: Calculate the training loss $L_{train}$ and update $\mathbf{W}$.
16: Calculate the operation weights $\mathbf{C}$ on top of $\alpha$.
17: Calculate the validation loss $L_{valid}$ and update $\alpha$.
18: Preserve the operation with the largest weight in each mixed operation.
19: return The searched architecture.

A OPTIMIZATION

In this paper, we optimize the supernet parameters $\alpha$ and operation parameters $\mathbf{W}$ with gradient descent as shown in Alg. 1. In the unified framework, the output $\mathbf{H}^{output}$ can be calculated as shown in Line 4-14. The mixed selection operation $f^i_j$ and fusion operation $f^j_i$ are given in Eq. (1) and Eq. (2), respectively. For the learnable aggregation operation, the mixed aggregation operation in block $i$ can be calculated as $f^i_j(\mathbf{H}, \mathbf{A}) = \sum_{k=1}^{N} c^i_k o^j_k (\mathbf{H}, \mathbf{A})$. For the predefined aggregation, $f^j_i(\mathbf{H}, \mathbf{A})$ equals to the results of the predefined operation. After obtaining the output $\mathbf{H}^{output}$, the cross-entropy loss $L_{train}$ and $L_{valid}$ can be generated and we can update the parameters with gradient descent as shown in Alg. 1.

B DETAILS OF EXPERIMENTS

B.1 Datasets

Cora [38], DBLP [2] and PubMed [38] are citation networks where each node represents a paper, and each edge represents the citation relation between two papers; Computers [32] is the Amazon co-purchase graph where nodes represent goods that are linked by an edge if these goods are frequently bought together; Physics [39] is a co-authorship graph where nodes are authors who are connected by an edge if they co-author a paper; Actor [34] is a co-occurrence graph where each node correspond to an actor and each edge denotes co-occurrence of these two actors on the same Wikipedia page; Wisconsin [34] is the hyperlinked web page graph where nodes represent web pages and edges are hyperlinks between them; In Flickr [53], nodes represent images and edges represent two images that share some common properties (e.g., same geographic location, same gallery, comments by the same user, etc.).

B.2 Baselines

To make a fair comparison with F^2SAGE and F^2GAT, we provide nine GNNs constructed with the predefined aggregation operation and different topology designs. For two stacking baselines, RES and JK methods, the topology designs of these methods are shown in Figure 3. The topology design of the DENSE method is constructed based on DenseGCN [25]. To be more specific, the aggregation operations are densely connected and the fusion operation CONCAT is utilized in each layer.

For GCNII, the aggregation operation can be represented by $H^{[+]} = \sigma((1 - \alpha_i)\mathbf{PH}^{i} + \alpha_i(\mathbf{H}^{i} + \mathbf{I} \beta_i)\mathbf{I} + \beta_i \mathbf{W}^{i})$. $\alpha_i$ and $\beta_i$ are the hyperparameters, $\mathbf{P} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$ is the graph convolution matrix with the renormalization trick. For simplicity, we ignore the feature transformation thus the feature fusion strategy in GCNII can be represented as $H^{[+]} = \alpha \mathbf{H}^{i} + (1 - \alpha)\mathbf{f}_o(\mathbf{H}^{i})$ where $\mathbf{f}_o$ is the aggregation operation. The illustration is shown in Figure 3(d). The GCNII baseline in our experiment is constructed on top of Figure 3(d) where we add the features of $\mathbf{H}^{0}$ and $\mathbf{H}^{i-1}$ in block $i$, then an aggregation operation or MLP followed behind.

In Figure 3(e), one layer PNA with four aggregation operations is provided. The PNA baseline in our experiment is constructed by stacking two layers with eight aggregation operations. In Figure 3(f), one layer MixHop with three aggregation operations is provided and the MixHop baseline in our experiment is constructed by stacking two layers.

We provide the performance comparisons of the GCNII, PNA and MixHop baselines used in our experiment and used in PyG 3.

As shown in Table 7, our baselines can achieve considerable performance on top of the unified framework with the same evaluation stage which will be introduced in the following.

Compared with F^2GNN, we provide the SNAG and SANE baselines. As shown in Figure 8, they focus on designing the aggregation operations in each layer, connections and layer aggregations in the output block. The search spaces are shown in Table 6.

3https://github.com/pyg-team/pytorch_geometric/tree/master/examples
Table 6: The search spaces in SANE and SNAG.

| Aggregation                  | SANE                        | SNAG                        |
|------------------------------|-----------------------------|-----------------------------|
| SAGE-SUM, SAGE-MEAN, SAGE-MAX-GCN, GAT, GAT-SYM, GAT-COS, GAT-LINEAR, GIN, GeniePath | GCN, SAGE-SUM, SAGE-MEAN, SAGE-MAX, SAGE-MAX-GCN, GCN, SAGE-SYM, SAGE-MEAN, GAT-SYM-LINEAR, GAT-COS-LINEAR, GIN, GeniePath |
| Skip-connection              | ZERO, IDENTITY              | ZERO, IDENTITY              |
| Layer aggregation            | CONCAT, MAX, LSTM           | CONCAT, MAX, LSTM           |

B.3 Implementation details

All models are implemented with Pytorch [33] on a GPU 2080Ti (Memory: 12GB, Cuda version: 10.2). Thus, for consistent comparisons of baseline models, we use the implementation of all GNN baselines by the popular GNN library: Pytorch Geometric (PyG) (version 1.6.1) [14] 4.

For SNAG [58] 5, an RL based method to design the aggregation operations in each layer, connections and layer aggregations in the output block, we use the 4-layer backbone to make a fair comparison with the proposed methods. In the search process, we set the search epoch to 500. In each epoch, we sample 10 architectures and use the validation accuracy to update the controller parameters. After training finished, we sample 5 candidates with the controller.

For SANE [59] 6, a differentiable method to design aggregation operations in each layer, connections and layer aggregations in the output block, the 4-layer backbone is utilized. In the search process, we set the search epoch to 30. One candidate GNN can be obtained after the search process. Repeat 5 times with different seeds, we can get 5 candidates.

For each random baseline in Table 3, we randomly sample 100 GNNs from the designed search space and train these methods from scratch. 1 candidate GNN is derived based on the validation accuracy.

For our method, we set the search epoch to 400 and \( \lambda \) to 0.001. One candidate GNN can be obtained after the search process. Repeat 5 times with different seeds, we can get 5 candidates.

The searched GNNs and all human-designed baselines are fine-tuned individually with the hyperparameters shown in Table 8. Each method owns 30 hyper steps. In each hyper step, a set of hyperparameters will be sampled from Table 8 based on Hyperopt 7, and then generate final performance on the split data. We choose the hyperparameters for each candidate with the validation accuracy, and then select the candidate for SNAG, SANE and the proposed method with the validation accuracy.

After that, we report the final test accuracy and the standard deviations based on 10 repeat runs.

B.4 Searched Topologies

For sake of space, we only show the searched topologies of F^2SAGE and F^2GAT. The results of F^2GNN are shown in Figure 9.

https://github.com/pyg-team/pytorch_geometric
https://github.com/AutoML-Research/SNAG
https://github.com/AutoML-Research/SNAE
https://github.com/hyperopt/hyperopt

Figure 9: The searched topologies on Cora, Physics and Actor datasets with F^2GNN.

Figure 10: Comparisons of test accuracy and the MAD value on the Cora dataset. Predefined aggregation operations are the GraphSAGE and GAT. “L4” represents the 4 layer baseline (\( N = 4 \) in our method), and so on. Darker colors mean larger values.
Table 7: Performance comparisons of the baselines used in our experiment and used in PyG. We report the average test accuracy and the standard deviation with 10 splits. “L2” and “L4” mean the number of layers of the base GNN architecture, respectively. “OOM” means out of memory.

| Topology | Cora | DBLP | PubMed | Computer | Physics | Actor | Wisconsin | Flickr |
|----------|------|------|--------|----------|---------|-------|-----------|--------|
| SAGE     |      |      |        |          |         |       |           |        |
| GNNII (L4) | 85.83(0.42) | 84.46(0.45) | 89.21(0.24) | 91.38(0.27) | 96.45(0.15) | 35.70(1.11) | 81.57(4.13) | 52.24(0.29) |
| PNA (L2)  | 84.29(0.67) | 82.76(0.42) | 89.25(0.26) | 90.67(0.42) | 96.32(0.10) | 33.89(2.68) | 75.29(4.46) | 52.09(0.73) |
| MixHop (L2) | 84.81(0.95) | 82.65(0.65) | 89.25(0.28) | 89.57(0.24) | 96.11(0.17) | 35.19(0.62) | 81.57(2.51) | 51.75(0.59) |
| GAT      |      |      |        |          |         |       |           |        |
| GNNII (L4) | 84.50(1.06) | 83.83(0.33) | 88.44(0.25) | 91.91(0.11) | 96.14(0.15) | 30.29(0.78) | 55.29(6.25) | 53.03(0.29) |
| PNA (L2)  | 85.06(0.72) | 83.46(0.47) | 86.81(0.47) | 90.84(0.24) | 95.85(0.18) | 28.14(1.99) | 47.69(5.12) | 54.02(0.33) |
| MixHop (L2) | 85.38(1.04) | 82.50(0.34) | 88.91(0.19) | 91.27(0.37) | 96.46(0.21) | 35.70(0.90) | 81.57(4.40) | 53.67(0.30) |
| PyG      |      |      |        |          |         |       |           |        |
| GCNII (L4) | 83.27(1.14) | 82.78(0.56) | 89.03(0.22) | 42.08(3.07) | 96.08(0.07) | 32.16(2.83) | 67.25(7.89) | 42.91(0.82) |
| PNA (L2)  | 86.99(0.57) | 84.05(0.18) | 88.99(0.15) | 91.61(0.11) | 96.60(0.11) | 33.43(1.35) | 68.24(6.00) | 52.18(0.17) |
| MixHop (L2) | 86.51(0.55) | 82.24(0.50) | 88.58(0.38) | 90.57(0.33) | 96.53(0.14) | 36.89(0.77) | 78.04(4.19) | OOM    |

Table 8: Hyperparameters we used in this paper.

| hyperparameter | Operation               |
|----------------|------------------------|
| Embedding size | 16, 32, 64, 128, 256, 512 |
| Learning rate  | [0.001, 0.01]          |
| Dropout rate   | 0, 0.1, 0.2, · · ·, 0.9 |
| Weight decay   | [0.0001, 0.001]        |
| Optimizer      | Adam, Adagrad          |
| Activation function | ReLU, ELU               |

Table 9: Statistics of the six heterophily datasets.

| Datasets | #Nodes | #Edges | #Features | #Classes | h  |
|----------|--------|--------|-----------|----------|----|
| Texas    | 183    | 309    | 1703      | 5        | 0.11 |
| Wisconsin| 251    | 466    | 1,703     | 5        | 0.21 |
|Actor     | 7,600  | 30,019 | 932       | 5        | 0.22 |
|Squirrel  | 5,201  | 198,493| 2,089     | 5        | 0.22 |
|Chameleon | 2,277  | 31,421 | 2,325     | 5        | 0.23 |
|Cornell   | 183    | 280    | 1,703     | 5        | 0.3  |

B.5 Heterophily results

B.5.1 Datasets. As shown in Table 9, Actor [34] is a graph representing actor co-occurrence in Wikipedia pages, Texas, Wisconsin and Cornell are the hyperlinked web pages of various universities provided by [34], Chameleon and Squirrel are the subgraphs of web pages in Wikipedia discussing the corresponding topics provided by [37].

B.5.2 Performance comparisons. On the heterophily datasets where the connected nodes are from different classes in most cases, the general aggregation operations cannot capture enough information from the graph structure and the MLP can even outperform several existing baselines as shown in H$_2$GCN[60]. It proved that the features generated in the intermediate layers help to increase the representation power of the model under heterophily, and the performance comparisons results in Table 10 can demonstrate the effectiveness of this method. Compared with H$_2$GCN which selected features in the output block, our method utilizes different levels of features in each block which has a more flexible feature utilization stage than H$_2$GCN. The top ranking we achieve can demonstrate the effectiveness of the feature fusion method in heterophily settings.

B.6 Alleviating the over-smoothing problem

For sake of space, we only provide 4 baselines in Figure 5, and the complete results are shown in Figure 10.
Table 10: Performance comparisons on the heterophily datasets. The “∗” results of all these methods are obtained from [60]. The best results in this table are highlighted in gray, and the second results are underlined. We provide the average accuracy rank as well and highlight the Top-3 methods in this table.

| Method         | Texas        | Wisconsin    | Actor         | Squirrel      | Chameleon     | Cornell       | Avg. Rank |
|----------------|--------------|--------------|---------------|---------------|---------------|---------------|-----------|
| F^0SAGE        | 0.8405^0.0197| 0.8588^0.0092| 0.3661^0.0109| 0.3604^0.0148| 0.5864^0.0186| 0.8324^0.0037| 6.5       |
| F^0GCN         | 0.8378^0.0527| 0.8609^0.0571| 0.3701^0.0101| 0.4184^0.0157| 0.6331^0.0086| 0.8352^0.0066| 3.5       |
| F^0GAT         | 0.8270^0.0599| 0.8706^0.0413| 0.3665^0.0117| 0.4732^0.0243| 0.6781^0.0200| 0.8351^0.0670| 2.5       |
| F^0GNN         | 0.8297^0.0514| 0.8824^0.0527| 0.3708^0.0100| 0.3658^0.0167| 0.6757^0.0216| 0.8270^0.0677| 4.17      |
| H^2GCN-1∗      | 0.8456^0.0087| 0.8667^0.0247| 0.3586^0.0111| 0.3642^0.0139| 0.5711^0.0159| 0.8216^0.0080| 6.67      |
| H^2GCN-2∗      | 0.8245^0.0128| 0.8588^0.0422| 0.3562^0.0130| 0.3790^0.0202| 0.5939^0.0198| 0.8216^0.0690| 7.67      |
| GraphSAGE+     | 0.8248^0.0614| 0.8119^0.0556| 0.3423^0.0099| 0.4161^0.0074| 0.5873^0.0168| 0.7595^0.0301| 8.5       |
| GCN-Cheby+     | 0.7789^0.0407| 0.7941^0.0446| 0.3411^0.0109| 0.4386^0.0146| 0.5524^0.0276| 0.7432^0.0746| 10.5      |
| MixHop         | 0.7786^0.0733| 0.7588^0.0490| 0.3222^0.0234| 0.4386^0.0148| 0.6050^0.0253| 0.7351^0.0634| 9.83      |
| GraphSAGE+JK+  | 0.8378^0.0024| 0.8196^0.0435| 0.3428^0.0101| 0.4085^0.0125| 0.5811^0.0197| 0.7568^0.0103| 8.17      |
| Cheby+JK+      | 0.7838^0.0637| 0.8255^0.0457| 0.3514^0.0137| 0.4503^0.0173| 0.6379^0.0227| 0.7459^0.0787| 6.83      |
| GCN+JK+        | 0.6649^0.0664| 0.7431^0.0463| 0.3418^0.0085| 0.4045^0.0161| 0.6342^0.0200| 0.6459^0.0868| 10.3      |
| GCN+           | 0.5946^0.0045| 0.5980^0.0099| 0.3026^0.0079| 0.3687^0.0134| 0.5982^0.0228| 0.5703^0.0286| 13.3      |
| GAT+           | 0.5838^0.0445| 0.5524^0.0087| 0.2629^0.0173| 0.3062^0.0211| 0.5469^0.0185| 0.5892^0.0315| 15.5      |
| GEOM-GCN+      | 0.6537        | 0.6412        | 0.3163        | 0.3814        | 0.609         | 0.6081        | 11.83     |
| MLP+           | 0.8189^0.0278| 0.8529^0.0681| 0.3576^0.0088| 0.2968^0.0181| 0.4636^0.0252| 0.8108^0.0033| 10.17     |