Feature Correlation Aggregation: on the Path to Better Graph Neural Networks

1st Jieming Zhou  
*College of Engineering and Computer Science*  
*Australian National University*  
Canberra, Australia  
jieming.zhou@anu.edu.au

2nd Tong Zhang  
*Image and Visual Representation Laboratory*  
*EPFL*  
Lausanne, Switzerland  
tong.zhang@epfl.ch

3rd Pengfei Fang  
*College of Engineering and Computer Science*  
*Australian National University*  
Canberra, Australia  
pengfei.fang@anu.edu.au

4th Lars Petersson  
*Imaging and Computer Vision Group*  
*CSIRO*  
Canberra, Australia  
Lars.Petersson@data61.csiro.au

5th Mehrtash Harandi  
*Department of Electrical and Computer Systems Engineering*  
*Monash University*  
Melbourne, Australia  
mehrtash.harandi@monash.edu

**Abstract**—Prior to the introduction of Graph Neural Networks (GNNs), modelling and analyzing irregular data, particularly graphs, was thought to be the Achilles’ heel of deep learning. The core concept of GNNs is to find a representation by recursively aggregating the representations of a central node and those of its neighbor, and many GNNs’ designs have demonstrated its success. However, most only focus on using the first-order information between a node and its neighbors. In this paper, we introduce a central node permutation variant function through a frustratingly simple and innocent-looking modification to the core operation of a GNN, namely the Feature cOrrelation aGgregation (FOG) module, which learns the second-order information from feature correlation between a node and its neighbors in the pipeline. By adding FOG into existing variants of GNNs, we empirically verify this second-order information complements the features generated by original GNNs across a broad set of benchmarks. A tangible boost in the model’s performance is observed where the model surpasses previous state-of-the-art results by a significant margin while employing fewer parameters. e.g., 26.202% improvement on a real-world molecular dataset using graph convolutional networks.

**Index Terms**—GNN, representation learning, classification, regression, few-shot

I. INTRODUCTION

Deep learning, especially in the form of convolutional neural networks (CNNs), has achieved tremendous successes in various machine learning tasks, such as image classification [1], object detection [2] and machine translation [3] etc. However, it remains a big challenge dealing with non-grid or irregular data such as protein-interaction networks, social networks, and knowledge graphs that may best be considered graph-structured data. Due to abundant graph-structured data, graph neural networks (GNNs) have been attracting increasing attention and have successfully been applied to many tasks. Consequently, we can see in the literature that much research has gone into investigating deep learning architectures and finding powerful representations for such graph-structured data.

On the path to finding powerful and discriminative representations, exploiting pairwise relationships within graph data and their feature vectors in a principled way has become a pivotal part of the graph learning area. One of the earliest works can be traced back to [4] in 2015, and the following works, including Chebyshev Convolutional Neural Networks (ChebyNets) [5], Graph Convolutional Neural Networks (GCNs) [6], GraphSAGE [7] and Graph Attention Networks (GATs) [8] demonstrate the importance of leveraging neighborhood information. Although those methods are derived from different perspectives (the spatial and spectral domain), all of them follow the message-passing scheme where messages between connected nodes are iteratively passed and permutation-invariant aggregation functions (such as mean, summation, and maximum) are employed in each layer to learn a representation for each node or graph.

Recent work [9] proposes a simple architecture, GIN, which is as powerful as Weisfeiler-Lehman (WL) [10]. Meanwhile, they unveil that a GNN can achieve this when it learns an injective multiset function consisting of a summation aggregator [9]. A multiset, mathematically speaking, denotes a set of feature vectors of a node and its neighbors, while nodes’ class and their feature vectors can be repeated. GNNs with mean or maximum aggregators will fail to learn an injective multiset...
function in this scenario. However, during the training stage, a GNN may still generate indiscriminative features after the summation aggregator when the network has not converged to an injection function, harming the performance.

Moreover, by gathering information layer by layer, all the nodes in a multiset tend to have similar representations, also known as over-smoothing. This is also the reason why these types of works are favoured by assortative graphs such as citation networks but degenerate significantly in disassortative graphs where the nodes of the same class share a high structural similarity but are far apart from each other. Those node sets are known as over-smoothing. This is also the reason why these types of works are favoured by assortative graphs such as citation networks but degenerate significantly in disassortative graphs where the nodes of the same class share a high structural similarity but are far apart from each other.[11]. This phenomenon is consistent with the most recent paper[12], which thoroughly demonstrates the performance of different architectures on various benchmarks.

The core part of GNNs is learning a parametrized network that works as an injection function to project every multiset into a unique representation[9]. Existing approaches ignore that, given the same multiset, two central nodes might have significantly different second-order feature spaces while sharing the same first-order feature space. Hence, we propose a novel central node permutation variant aggregation module called Feature Correlation Aggregation (FOG) to discover the intrinsic representation ability of a multiset. To overcome the limitations of current GNNs, we have employed the Kronecker product of a multiset’s feature vector to create a correlation space that leverages second-order statistical information. This approach enables us to address the aforementioned drawbacks more effectively and efficiently. In this new space, we aggregate the correlation feature vectors as complementary information based on the aggregation features of existing nodes. Despite the two multisets being quite similar, the resulting feature extracted from the second-order space is distinctive, while the simple first-order aggregation fails to find the difference. In addition, our FOG may be readily layered on top of any existing GNNs. As Fig. 1 shows, FOG helps the GateGCN-E [13] to learn more discriminative features and achieve lower cross-entropy loss compared to the original networks.

In a nutshell, our contribution to this paper is three-fold.

1) We discuss and analyze the weaknesses in current GNN structures while proposing a new FOG module that dynamically aggregates feature representations from the feature correlation space for each multiset.

2) Our FOG module is extremely simple and can be inserted in any GNN structure. We carefully design and embed our FOG module into GCNs, GAT, GatedGCNs, GIN, and GraphSAGE with fewer parameters than the original models to provide fair comparisons.

3) We conduct comprehensive experiments on graph pattern recognition, node classification, graph regression, and edge classification on different datasets to validate our module’s effectiveness and compatibility.

II. RELATED WORKS

Graph Neural Networks. GNNs were first proposed by Gori[14] and have since evolved and been applied in a much wider range of applications. Most GNNs are considered from either a spectral or a spatial perspective. For example, by defining the Laplacian matrix for a given graph, the orthogonal graph transform basis can be obtained by applying SVD on the Laplacian matrix. Spectral-based methods can filter the feature vectors using the element-wise Hadamard product in the spatial domain, equivalent to applying the convolution operation in the graph domain. ChebyNets[5] approximates the k-th order convolutional kernel by recursively employing Chebyshev polynomials. GCNs[6] replace the k-th order Chebyshev expansion with a first-order approximation to simplify the structure.

From the spatial perspective, existing methods focus on picking appropriate neighbors during aggregation. GraphSAGE[7] proposes a sampling algorithm that randomly samples a fixed number of neighbors during training. FastGCN[15] and AdaptGCN[16] propose to apply different sampling methods in the local neighborhood to alleviate the exponential growth in the number of locally sampled nodes and thereby speed up the convergence of the optimization. Instead of dropping nodes, Graph Attention Network (GAT)[8] adopts a multi-head attention mechanism[3] from an NLP task to learn a weight for each central-neighbor pair.

Feature Correlation Description. Feature correlation has been studied extensively for visual tasks, such as image or object retrieval and detection. The Kronecker product is the most widely used operation for capturing second-order statistical information[17]–[24]. In detection tasks, covariance matrices are investigated to represent the regional descriptors[17],[18]. Lin et al. proposed the bilinear CNN where the outer product of two feature vectors, generated from two non-identical networks, is used to model the local pairwise interactions[19]. A further study in[21] proved that obtaining rich correlation statistics can also be achieved by taking the
outer product of a feature vector with itself. Cross-modality features, e.g., visual and language, can also be fused via a bilinear operation [20]. In [23], second-order auto-correlation, generated via a bilinear mapping, is used in an attention block to identify salient regions within the images.

In this paper, the proposed FOG module extracts a more discriminative feature from a correlation space constructed using the Kronecker product between the central node and its neighbours in a graph neural network. This simple yet effective module brings significant performance gains across various tasks on graph-structured data over and above existing state-of-the-art GNNs.

### III. METHODOLOGY

#### A. Notation

We show a graph $G$ with nodes $V = \{v_1, v_2, \ldots, v_n\}$ and edge set $E \subseteq V \times V$ by $G = (V, E)$. For each node $v \in V$, its feature representation is denoted by $\tilde{h}_v \in \mathbb{R}^{C_v}$, where $C_v$ is the channel dimension of the node feature. Given $v$ as a central node, its neighbour set is defined as $\mathcal{N}(v) = \{u_i \in V : (v, u_i) \in E, i = 1, 2, \ldots, |\mathcal{N}(v)|\}$. The cardinality of the neighbour set of $v$ is shown by $|\mathcal{N}(v)|$. The feature of the edge connecting node $v$ and $u_i$ is denoted by $e_{uv_i} \in \mathbb{R}^{C_e}$, where $C_e$ is the channel dimension of the edge feature. Other notations used in this paper are illustrated in Table I.

Formally, the $l$-th layer/iteration layer of a GNN can be formulated as,

$$ a^l_v = \text{AGGREGATE}(\{ h^{l-1}_u : u \in \mathcal{N}(v) \} ) $$

$$ h^l_v = \text{COMBINE}(h^{l-1}_v, a^l_v) $$

where $\text{AGGREGATE}(\cdot)$ is an aggregation function that aggregates features of neighbors, and $\text{COMBINE}(\cdot)$ means the combination of a central node and its neighbors’ aggregated features [9]. Note that most GNNs propose a modification of the AGGREGATE and COMBINE functions, which only obtains the first-order features, e.g., in GAT, AGGREGATE is a non-linear function, in which neighbors’ features are aggregated by weighted summation. Meanwhile, GIN directly uses a summation aggregator for neighbors but places a learnable weight on the central node and a non-linear function in the COMBINE. In contrast, we are taking the feature correlation of the central node and its neighbors into account, and this second-order information can be added on top of any existing GNN as follows:

$$ h^l_v = \text{Concat}(\text{COMBINE}(h^{l-1}_v, a^l_v), f_{\text{FOG}}(h^{l-1}_v, \{h^{l-1}_{u_i}\})) $$

where the $f_{\text{FOG}}(\cdot, \cdot)$ is our FOG module and will be described in detail in the following section. In Equation 3, our module works in a different path from the previous methods. In this scenario, FOG generates a more discriminative and diversified feature vector that describes the correlation information between the central node and its neighbors as Figure 2 shows. This mechanism enables FOG to introduce complementary information to existing GNN architectures, prevent the GNNs from over-smoothing to some extent, and be a powerful independent architecture.

#### B. FOG module

Let the feature vector of a node $v$ in layer $l-1$ be $h^{l-1}_v \in \mathbb{R}^{C_{vl}}$. The neighbors of $v$, shown by $u_i \in \mathcal{N}(v)$, form a neighbor feature-set $\mathbb{H}_{\mathcal{N}(v)}^{l-1} = \{ h^{l-1}_{u_i} \}$, where $h^{l-1}_{u_i} \in \mathbb{R}^{C_{u_i}}$. We first perform a non-linear mapping on $h^{l-1}_v$ and $h^{l-1}_{u_i}$ to a possibly lower-dimensional space via $W_v^l \in \mathbb{R}^{C_{vl} \times C_{vl}}$ and a ReLU function. Applied by another mapping $W_{u_i}^l \in \mathbb{R}^{C_{u_i} \times C_{u_i}}$ with ReLU, the dimensions of neighbor features are further reduced as:

$$ \hat{h}_v^l = \text{BN}(\text{ReLU}(W_v^l h^{l-1}_v)) $$

$$ \hat{h}_{u_i}^l = \text{BN}(\text{ReLU}(W_{u_i}^l \text{BN}(\text{ReLU}(W_v^l h^{l-1}_v)))) $$

This is followed by computing the Kronecker product between the central node representation (i.e., $h_v^l$) and the corresponding neighbor nodes (i.e., $h_{u_i}^l$) as follows:

$$ \mathbb{R}^{C_{vl} \times C_{u_i}} = \mathcal{K}(\hat{h}_v^l, \hat{h}_{u_i}^l) = \hat{h}_v^l \otimes \hat{h}_{u_i}^l $$

In essence, the Kronecker product on $\hat{h}_v^l$ and $\hat{h}_{u_i}^l$ will enable us to benefit from the correlation information among the central node $v$ and its neighbors. We aggregate the correlation among all the neighbors using $f_{\text{agg}}(\cdot)$ (i.e., summation) into a single vector of dimensionality $\mathcal{C}_{vl} \times \mathcal{C}_{hl}$, as:

$$ f_{\text{agg}}(\{h^{l}_{u_i}\}) = \sum_{i=1}^{N} f_{\text{agg}}(h_v^l, h_{u_i}^l) $$
When aggregating neighbors’ node features to a central node, GNN only passes the first-order information in the multiset. However, FOG can pass the second-order information through the Kronecker Product operation. Note that a non-linear layer is applied for dimension reduction after the Kronecker Product operation.

Finally, we pass the aggregated information to a linear mapping $W_{l}^{vu} \in \mathbb{R}^{C_{p} \times C_{h1} \times C_{h2}}$ and obtain the output of the FOG module as

$$p_l^v = f_{\text{FOG}}(h_{l}^{v-1}, \{h_{u_i}^{l-1}\}) = W_{l}^{vu}f_{\text{agg}}(\{\tilde{h}_{u_i}^{l}\}). \quad (8)$$

For a plain FOG model, the final output $h_{l}^{v} \in \mathbb{R}^{C_{q}}$ is obtained as:

$$h_{l}^{v} = \text{ReLU}(p_l^v). \quad (9)$$

The FOG has the ability to improve a base GNN module by adding additional second-order information once combined with it. Given a base GNN module with functionality $f_{\text{base}}$, we can also attain a representation for node $v$ as

$$\mathbb{R}^{C_{q}} \ni q_l^v = f_{\text{base}}^{v}(h_{l}^{v-1}, \{h_{u_i}^{v-1}\}). \quad (10)$$

After concatenating $p_l^v$ and $q_l^v$ along the channel, and subsequently passing the result to an activation function $\phi(\cdot)$ that is used in the original base GNN, the final output $h_{l}^{v} \in \mathbb{R}^{C_{\text{out}}}$ is obtained as:

$$h_{l}^{v} = \phi(\text{Concat}(p_l^v, q_l^v)), \quad (11)$$

where $C_{\text{out}} = C_{p} + C_{q}$.

As Equation 11 shows, with a minimal amount of changes by concatenating $p_l^v$ and $q_l^v$, the FOG module can be integrated into most existing GNNs thereby introducing correlation information boosting the performance. In this paper, we introduced the FOG module into five state-of-the-art modules, including GCN, GAT, GatedGCN, GIN, and GraphSAGE. Furthermore, we also evaluate FOG with a 2-layer GNN as proposed by [25] on a few-shot learning task.

### IV. Experiments

In this section, most experiments are conducted across various datasets based on the benchmarking procedure proposed by [12]. This benchmark covers most applications of GNNs, i.e., graph pattern recognition, semi-supervised graph clustering, graph regression, and edge classification. The five aforementioned GNN baselines reported by Dwivedi et al. are compared with their FOG counterparts, along with a FOG-only model, under similar training protocols as in [12]. By introducing FOG to a GNN-based few-shot learning problem proposed by Garcia and Bruna [25], we explore the possibility of applying FOG to a computer vision task. Furthermore, three more experiments are conducted in the ablation study and further discussion to demonstrate the parameter-reducing potentials and limitations of FOG.

### A. Datasets

**SBM PATTERN and CLUSTER.** We use the SBM datasets generated by the stochastic block model for graph pattern recognition and semi-supervised graph clustering tasks. These two datasets are proposed by [12]. The SBM PATTERN set aims to identify a fixed graph pattern embedded in a large graph. This set comprises 10K items in the train set, 2K in the validation and 2K in the test set. Each graph has, on average, 117.47 nodes and two classes. The SBM CLUSTER evaluates the model’s performance on a semi-supervised clustering task. Given one prior known node label per class, the model learns to gather nodes that belong to the same class according to their connectivity. The number of graphs is 10K for train, 2K in the validation and 2K in the test set. Each graph has, on average, 117.47 nodes and two classes. The SBM CLUSTER evaluates the model’s performance on a semi-supervised clustering task. Given one prior known node label per class, the model learns to gather nodes that belong to the same class according to their connectivity. The number of graphs is 10K for train, 1K for validation, and 1K for test in this dataset. The average number of nodes in each graph is 117.20, and the number of classes is six.

**ZINC.** The ZINC dataset is used to regress the molecular constrained solubility [26]. It contains 10K, 1K and 1K graphs.
in the train, validation and test set, respectively, with each graph containing 23.16 nodes on average. For each graph, the node features are the type of atoms, and the edge features are bonds.

**TSP.** We employ the TSP dataset generated by [12] to evaluate the edge classification performance of our method. The ability of the algorithm to solve NP-hard combinatorial optimization problems is evaluated on this dataset. The node feature represents the coordinates of a node in a unit square. The train, validation and test sets are split into 10K, 1K and 1K graphs, respectively. Each graph has a different number of nodes. The average number is 275.76.

**MiniImageNet.** In our experiments, miniImageNet proposed by [27] is used for a few-shot learning task. This dataset contains 100 classes and 600 images for each class. Following [28], we use a 64/16/20 split for training/validation/testing on 5-way 1-shot and 5-way 5-shot tasks.

**IMDB-MULTI.** IMDB-MULTI is a graph classification benchmark proposed by Yanardag and Vishwanathan [29]. In this social network dataset, each actor/actress’ ego-network forms a graph representing one of three genres. Actors/actresses are represented as nodes and connected if they performed in the same movie. There are 1500 graphs in this dataset.

**B. Implementation and Evaluation**

To verify the superior performance of our proposed FOG module, we evaluate FOG against various baselines mentioned in § III-B. Empirical results reveal that the FOG module significantly gains performance over the baseline GNNs with fewer parameters. This observation suggests that the FOG module benefits from the feature generated by the correlation feature space and non-linearity.

The GNN baseline and its FOG counterpart share the same training protocol for each dataset. To thoroughly verify the effectiveness of our proposed method, if not otherwise mentioned, we used a grid search to find the optimized learning rate in \{1e^{-2}, 5e^{-3}, 1e^{-3}, 5e^{-4}\} and weight decay in \{1e^{-3}, 1e^{-6}, 0\}, and evaluated the network ten times using different random seeds, more than that in [12]. The Adam optimizer is used in all tasks with \(\beta_1 = 0.9\), and \(\beta_2 = 0.999\). To verify that the performance gain comes from our proposed algorithm, the number of parameters of the FOG-only model and FOG-equipped GNN is similar to the baseline GNN. All GNNs evaluated on [12] use residual connections [30] and batch normalization [31]. We omit the self-loop in the input graphs across all experiments. We use the PyTorch [32] deep learning package to implement our algorithm based on the code provided by [12]. All experiments are trained on an Nvidia Tesla V100 16GB GPU. By default, two layers of a GNN module are used on miniImageNet and four layers are used on the other datasets. The detailed architectures and hyper-parameter settings are provided in the supplementary material.

**SBM PATTERN and CLUSTER.** The patience value is set to five. After five epochs with no improvement of the loss, the learning rate will be reduced by a factor of two. The training progress will stop when the learning rate is smaller than \(1e^{-5}\). All final node features are passed to a 3-layer multilayer perceptron (MLP), including a classification layer that uses the cross-entropy loss to obtain a prediction for each node. The performance metric is the average accuracy between predicted and ground-truth labels.

The evaluation of the SBM dataset is shown in Table II and III. In the graph pattern recognition task, the model with only the FOG module surpasses all architectures by a maximum of 21.783% in terms of accuracy when compared to GCN (see Table II). Introducing FOG to GCN, GAT, GatedGCN, and GraphSAGE boosts the original structures. However, a performance drop is observed when combining FOG with GIN. We conjecture that how GIN adds the learnable weights in the COMBINE part conflicts with our FOG.

In the semi-supervised graph clustering task, our FOG module also shows its potential to bring performance gains across different baselines (see Table III). Furthermore, plugging in the FOG module consistently improves the accuracy across all baselines and pushes them even higher than the FOG-only and corresponding baseline models. The FOG module improves the accuracy of GCN, GAT, GatedGCN, GIN, and GraphSAGE by 7.334%, 3.089%, 4.079%, 3.624%, and 9.689%, respectively.

**TABLE II**

| Model       | #Param | Acc(\% \pm s.d.)         |
|-------------|--------|--------------------------|
| FOG         | 99,046 | \(85.663 \pm 0.025\)     |
| GCN         | 100,923| \(63.880 \pm 0.074\)     |
| GCN+FOG     | 101,026| \(85.663 \pm 0.027\)     |
| GAT         | 109,939| \(75.924 \pm 1.23\)      |
| GAT+FOG     | 101,346| \(85.654 \pm 0.23\)      |
| GatedGCN    | 104,003| \(84.480 \pm 0.122\)     |
| GatedGCN+FOG| 102,050| \(85.454 \pm 0.099\)     |
| GIN         | 100,884| \(85.590 \pm 0.041\)     |
| GIN+FOG     | 99,234 | \(85.524 \pm 0.029\)     |
| GraphSAGE   | 101,739| \(50.516 \pm 0.001\)     |
| GraphSAGE+FOG| 95,679 | \(85.578 \pm 0.061\)     |

ZINC. The same training protocol for SBM is used here, and the patience value is set to 10. Instead of classification, we target a regression task in ZINC, where a 3-layer MLP is followed after the GNN’s last layer to approximate the ground truth. The mean absolute error (MAE) between the predicted and ground-truth constrained solubility is applied as a loss function and the performance metric. In the graph regression task, our FOG module improves over most baseline GNNs for the MAE value by a large margin, shown in Table IV. In particular, the GCN+FOG model improves 26.202% over GCN and exceeds all baselines. Considering that the FOG-only model performs less than most FOG-equipped baseline GNNs, the FOG module can provide more complementary information to baseline GNNs on this dataset. However, introducing FOG to GIN causes a performance drop, similar to SBM PATTERN.
TSP. The following changes are made when adapting the SBM training protocol: (a) The patience value is set to 10. (b) The F1 score only considers the positive class for the performance metric. (c) Edge features are passed to a 3-layer MLP whose last layer is a classification layer. Cross-entropy loss is used for this edge classification task.

The edge classification task also benefits from our FOG module. As shown in Table V, our FOG module consistently improves over the baseline GNNs. For example, the GCN+FOG improves 9.206% F1 over GCN. In addition, the best performance architecture, GatedGCN+FOG, outperforms the state-of-the-art baseline (i.e., GatedGCN) by 2.351% F1 score when using edge features.

MinImageNet. The experiments on this dataset are based on the networks proposed by [25]. One query image and its support set form a fully-connected graph where each node is an image. Each architecture has two parts: (1) A 5-layer CNN generates a representation vector as a node feature. (2) A 2-layer GNN passes representations among nodes to predict the label of the query image among the images in the support set. We only concatenate FOG to the first GNN layer for the FOG-equipped version and replace the first GNN layer with FOG-equipped architecture: GatedGCN. Their FOG-equipped counterparts with FOG are trained four times with cross-entropy loss and four random runs.

C. Ablation Study

Dimensionality of hidden layers. According to the notable boost when introducing FOG into the state-of-the-art module, it demonstrates the ability of FOG to reduce the total number of parameters of the original architectures while still maintaining the performance. To demonstrate this point, we choose the simplest architecture, GCN, and the best-performing architecture: GatedGCN. Their FOG-equipped counterparts with different numbers of parameters are evaluated on ZINC by using the same training protocol as in § IV-B. All models use the same learning rate and weight decay found in previous experiments. Their parameters are roughly divided from 25% to 100% by adjusting hidden layer dimensions according to their baseline models.

As Table VII shows, all variations are better than their corresponding baseline models. For the best-performing baseline on ZINC, FOG boosts the performance by 11.733% compared to GatedGCN by only using about 25% of the parameters. Both 25% architectures are still better than the other baseline architectures.
D. Further Discussion

After demonstrating the strengths of our FOG, we address some limitations here. By leveraging the node’s feature, FOG gains substantial benefits. However, in the absence or lack of information from the node, FOG’s capabilities are significantly limited. Therefore, we do further study on the behaviour of our FOG on those datasets.

Other techniques, in comparison to ours, focus on graph structural information while ignoring the correlation on feature space. Among them, GIN and GatedGCN are the top two architectures on ZINC, regardless of whether they use edge features or not. Thus, we add another graph-structure-weighted dataset, IMDB-MULTI, with no node and edge features. We assess both of them and their FOG-equipped versions alongside an FOG-only design.

Following the setting of [9], we set all node features to the uninformative value one on the IMDB-MULTI, where 10-fold cross-validation and only one random seed is applied. The same patience value, learning rate reduction factor, and minimum learning rate are set as experiments on the SBM dataset, and a 3-layer MLP is used to predict labels using final node features. To show the importance of the nodes’ features, we replace the nodes’ feature vectors on the ZINC dataset with the uninformative value one on the IMDB-MULTI, where 10-fold cross-validation and only one random seed is applied.

Table VIII illustrates that compared to state-of-the-art models, the notable performance drops when only applying FOG on graphs without node feature. However, GatedGCN+FOG achieves better results than the baseline on both datasets. This implies FOG can still provide complementary information for the base GNNs while independent FOG may struggle to extract correlation information from graph structure. The consistent performance drop while concatenating FOG with GIN suggests that the learnable parameter $\epsilon$ in GIN may break the correlation between the central node and its neighbors, which makes these two modules incompatible in most scenarios. Generally, all networks suffer large performance drops compared to Table IV. This suggests original node features and edge features have important contributions to this kind of graphs.

V. Conclusion

We have presented a new aggregation module, FOG, which introduces the feature correlation between the central node and its neighbors into the central node’s feature representation. We first summarize the way existing GNNs work and generalize them into the same presentation. Distinguishing from existing methods, our method is compatible with most existing methods so that it can be inserted on top of those GNNs. Furthermore, our extensive experiments on different datasets have shown that FOG can help the GNNs get more discriminative features and perform better in various tasks, e.g., graph pattern recognition, node classification, graph regression, and edge classification. Furthermore, we also provide a thorough analysis of the limitations and applicability of our method. In the future, we will explore other types of functions sensitive to the change of central node and study how it affects the GNNs.

Acknowledgement. Tong Zhang was supported in part by the Swiss National Science Foundation via the Sinergia grant CRSII5-180359.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, “ImageNet: A Large-Scale Hierarchical Image Database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in 2014 European conference on computer vision, 2014.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in Advances in neural information processing systems, 2017.

[4] M. Henaff, J. Bruna, and Y. LeCun, “Deep Convolutional Networks on Graphs,” arXiv preprint arXiv:1506.05163, 2015.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering,” in Advances in Neural Information Processing Systems, 2016.

[6] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” in 5th International Conference on Learning Representations, ICLR 2017. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[7] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” in Advances in Neural Information Processing Systems, 2017.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” International Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=rJXmpkJZ

[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How Powerful are Graph Neural Networks?” in International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=ryGofiiA5km

[10] A. Leman and B. Weisfeiler, “A reduction of a graph to a canonical form and an algebra arising during this reduction,” Nauchno-Technicheskaya Informatsiya, vol. 2, no. 9, pp. 12-16, 1968.
[11] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, “struc2vec: Learning Node Representations from Structural Identity,” in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.

[12] V. P. Dwivedi, C. K. Joshi, A. T. Liu, T. Laurent, Y. Bengio, and X. Bresson, “Benchmarking graph neural networks,” Journal of Machine Learning Research, vol. 24, no. 43, pp. 1–48, 2023.

[13] X. Bresson and T. Laurent, “Residual Gated Graph ConvNets,” arXiv preprint arXiv:1711.07553, 2017.

[14] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in Proceedings of 2005 IEEE International Joint Conference on Neural Networks, 2005.

[15] J. Chen, T. Ma, and C. Xiao, “FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling,” in International Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=rYtstxWAW

[16] W. Huang, T. Zhang, Y. Rong, and J. Huang, “Adaptive Sampling Towards Fast Graph Representation Learning,” in Advances in neural information processing systems, 2018.

[17] O. Tuzel, F. Porikli, and P. Meer, “Region Covariance: A Fast Descriptor for Detection and Classification,” in Proceedings of the 9th European Conference on Computer Vision, 2006.

[18] O. Tuzel, F. Porikli, and P. Meer, “Human Detection via Classification on Riemannian Manifolds,” in 2007 IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[19] T. Lin, A. RoyChowdhury, and S. Maji, “Bilinear CNN Models for Fine-grained Visual Recognition,” in 2015 IEEE International Conference on Computer Vision, 2015.

[20] J.-H. Kim, J. Jun, and B.-T. Zhang, “Bilinear Attention Networks,” in Advances in Neural Information Processing Systems 31, 2018.

[21] T.-Y. Lin and S. Maji, “Improved Bilinear Pooling with CNNs,” in British Machine Vision Conference, 2017.

[22] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang, “End-to-End Deep Kronecker-Product Matching for Person Re-identification,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[23] P. Fang, J. Zhou, S. K. Roy, L. Petersson, and M. Harandi, “Bilinear Attention Networks for Person Retrieval,” in The IEEE International Conference on Computer Vision, 2019.

[24] P. Fang, J. Zhou, S. K. Roy, P. Ji, L. Petersson, and M. Harandi, “Attention in attention networks for person retrieval,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 9, pp. 4626–4641, 2022.

[25] V. G. Satorras and J. B. Estrach, “Few-shot learning with graph neural networks,” in International Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=BJj6qGbRW

[26] T. Sterling and J. J. Irwin, “Zinc15 – Ligand Discovery for Everyone,” Journal of Chemical Information and Modeling, 2015.

[27] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in Advances in Neural Information Processing Systems, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/90e135783365498361280533ec9148c-Paper.pdf

[28] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in ICLR, 2017.

[29] P. Yanardag and S. Vishwanathan, “Deep Graph Kernels,” in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’15, 2015, p. 1365–1374. [Online]. Available: https://doi.org/10.1145/2783258.2783417

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[31] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015.

[32] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable MPC for End-to-end Planning and Control,” in Advances in Neural Information Processing Systems, vol. 31, 2018, pp. 8289–8300. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/badbdd83eb42f14526c65d1807a9309-Paper.pdf