Enhanced Graph Neural Networks with Ego-Centric Spectral Subgraph Embeddings Augmentation

Anwar Said[†], Mudassir Shabbir[†§], Tyler Derr[†], Waseem Abbas[‡], Xenofon Koutsoukos[†]
[†]Vanderbilt University, USA
[‡]University of Texas at Dallas, USA
[§]Information Technology University, Pakistan

***Abstract*—Graph Neural Networks (GNNs) have shown remarkable merit in performing various learning-based tasks in complex networks. The superior performance of GNNs often correlates with the availability and quality of node-level features in the input networks. However, for many network applications, such node-level information may be missing or unreliable, thereby limiting the applicability and efficacy of GNNs. To address this limitation, we present a novel approach denoted as Ego-centric Spectral subGraph Embedding Augmentation (ESGEA), which aims to enhance and design node features, particularly in scenarios where information is lacking. Our method leverages the topological structure of the local subgraph to create topology-aware node features. The subgraph features are generated using an efficient spectral graph embedding technique, and they serve as node features that capture the local topological organization of the network. The explicit node features, if present, are then enhanced with the subgraph embeddings in order to improve the overall performance. ESGEA is compatible with any GNN-based architecture and is effective even in the absence of node features. We evaluate the proposed method in a social network graph classification task where node attributes are unavailable, as well as in a node classification task where node features are corrupted or even absent. The evaluation results on seven datasets and eight baseline models indicate up to a 10% improvement in AUC and a 7% improvement in accuracy for graph and node classification tasks, respectively.**

***Index Terms*—Graph Neural Networks, Subgraph Spectral Embeddings, Graph Descriptors, Abnormal Features**

I. Introduction

Graph representation learning has proved crucial to several real-world applications, including drug discovery & development [36], weather and traffic forecasting [11], recommendation in e-commerce [41], combinatorial optimization [4], etc. In the past few years, there has been a surge of interest in designing graph neural networks (GNNs), which are powerful tools for learning from graph-structured data [6], [15], [20]. GNNs have attained state-of-the-art performance on a variety of downstream Machine Learning (ML) tasks, such as node classification, graph classification, graph regression, and link prediction [30], [44]. For example, predicting the toxicity or property of molecules, item recommendations in an e-commerce website, and identifying users' communities in social networks [9], [42].

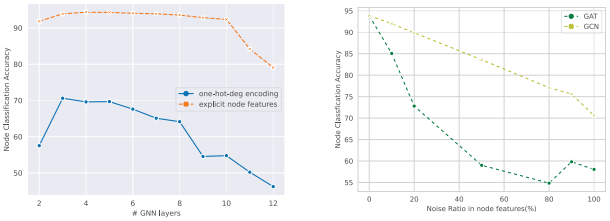

Figure 1: (a) GCN's performance on the Facebook dataset [27] using original features and one-hot-degree encoding. (b) performance with varying abnormality/noise ratios in the node features.

One key advantage of GNNs over manually engineered embeddings is the ability to learn a correlation between information specific to a node and its *global* position in the network. Explicit node features often play an integral role in the performance of GNNs since they encode valuable distinguishing criteria about the entities. For instance, in molecular network data, node features provide crucial information about the chemical nature of the element. Similarly, in citation networks, node features provide textual information on represented publications and contribute significantly to the model's overall performance. When this node-level information is unavailable, current methods use ad hoc techniques such as random vectors or vectors of ones. Numerous recent approaches also use one-hot degree encoding [43]. However, this is rarely a suitable substitute for the explicit node's features. In Figure 1 (a), we illustrate this by comparing the performance of explicit node features and one-hot-degree encoding on Graph Convolutional Networks (GCN) [20] with varying numbers of layers. We observe up to 25% improvement in the results for the explicit features compared to the degree encoding. These results evince that one-hot degree encoding significantly degrades the model's performance, necessitating the use of robust techniques to generate expressive node embeddings in graph lacking node features. Specifically, the design of a framework that produces topology-aware node features in situations when node features are unavailable is one goal of this study.

The node features could be missing or abnormal for a number of reasons, such as privacy concerns, human or machine error, adversarial error, and incomplete data entry [26]. For instance, in a social network, users may not have completed

Anwar Said, Tyler Derr and Xenofon Koutsoukos are with the Computer Science Department at the Vanderbilt University, Nashville, TN. Emails: {anwar.said,tyler.derr,xenofon.koutsoukos}@vanderbilt.edu.
Waseem Abbas is with the Systems Engineering Department at the University of Texas at Dallas, Richardson, TX. Email: waseem.abbas@utdallas.edu
Mudassir Shabbir is with the Computer Science Department at Information Technology University, Lahore, Pakistan and with Vanderbilt University, Nashville, TN, USA. Email: mudassir.shabbir@itu.edu.pk

1946-0759/23/$31.00 ©2023 IEEE
DOI 10.1109/ICMLA58977.2023.00341

their profile information, resulting in missing user features. Similarly, not all items in a co-purchase network may have a complete description associated with them. In a transportation network, traffic information coming from sensors may be noisy due to complex dynamics, leading to abnormal node features. We illustrated the effect, in Figure 1 (b) by running Graph Attention Network (GAT) [39], and GCN on varying ratios of random Gaussian noise in the node features on Facebook dataset [27]. We observe a significant decline in performance when noise is injected. Thus, the second objective of this study is to design a mechanism resilient to abnormal node features.

We observe that node features in many social networks originate the local subgraph topology. For example, keeping the number of followers as a node feature in a Facebook-page network reflects the in-degree of the node. At the same time, we can gather sufficient information of the graph's structure from its subgraphs. The reconstructibility conjecture, which holds true for many graph families, asserts that a graph can be exactly constructed from a collection of its (single) vertex-deleted subgraphs [1], [2]. A vertex-deleted subgraph for a vertex v is obtained by deleting v from the graph. Even more, if $B_k(v)$ is the set of vertices at most k-hop from v, and $G_k(v)$ is the subgraph of G obtained by deleting vertices in $B_k(v)$, then many graph properties can be inferred or a graph can be constructed exactly from a collection of such $G_k(v)$ [21], [23].

Therefore, we propose to use topology-aware subgraph embeddings that are based on the local topology of a network as features of a node. The proposed framework, denoted as Ego-centric Spectral subGraph Embeddings Augmentation (ESGEA) allows to design topology-aware node features using expressive graph embedding methods to enhance the existing node features. In applications where node features are unavailable, ESGEA provides a flexible way to produce node features that are expressive enough to obtain quality results. ESGEA consists of four modules: (a) ego-centric subgraph extraction, where $k-$hops local subgraphs are extracted for each node, (b), a spectral graph-embedding method is deployed to extract expressive graph representations on each subgraph, (c) feature augmentation module is proposed to augment features, and, (d) GNN learning module is provided to learn nodes/graph representations. The proposed approach is flexible to use any off-the-shelf graph-based embedding with any GNN-based architecture to design models for different applications. Unlike existing subgraph methods that learn node representations with nested subgraphs message passing, the proposed approach is novel in terms of bridging the gap with topology-aware graph descriptors to use with GNNs. Moreover, ESGEA provides a flexible learning framework that can be customized to meet the desired goal. We offer the following contributions:

- We introduce a topological feature augmentation method, Ego-centric Spectral subGraph Embedding Augmentation (ESGEA), that designs new or enhances corrupted/missing node features.
- We introduce a novel framework that offers flexibility for graph representation pipelines by combining spectral graph-based embeddings with GNNs based on ESGEA.
- We evaluate the proposed framework in graph and node classification settings where node features are unavailable or corrupted (e.g., in the presence of varying amounts of noise) and show its effectiveness through extensive experiments.

The structure of this paper is as follows. Section 2 presents an overview of the related work. Section 3 provides the preliminaries and an introduction to a few definitions, while Section 4 details the methodology of the proposed approach. In Section 5, an evaluation of the proposed approach in both graph and node classification setting is provided. Finally, Section 6 concludes the paper and outlines potential future directions.

II. Related Work

Graph Neural Networks (GNNs) have made substantial advancements in learning representations of graph-structured data in recent years [3]. GNNs essentially generalize end-to-end learning from regular grid data such as image, video, and text, to graph-structured data [42]. Unlike deep neural networks, the key idea behind such generalization is the message passing framework that smooths the message with respect to the local neighborhood [14]. The design of message passing are majorly motivated in spatial domain [10], [33] and spectral domain [8], [20]. GNNs literature broadly includes convolutional layers [20], [25], aggregation operators [15], pooling methods [45], and feature augmentation [5], [29].

There is a growing interest in minimizing the vulnerability of GNNs to node feature and graph structure noise in recent years. A few notable works in this direction include [7], [25], [47]. Similarly, numerous approaches for alleviating structural noise in GNN settings have been proposed, including [12], [46]. For further reading, please refer to the comprehensive survey of adversarial attacks on graphs [17], [18].

Unlike the existing techniques, we pursue a novel approach to advance graph learning in social networks through the incorporation of subgraph embeddings, with a primary focus on applications where crucial node features are absent. To this end, we have introduced a learning framework that integrates GNNs with graph descriptors, thereby presenting a comprehensive methodology that is adaptable to all types of graph embeddings and GNN architectures. Our evaluations in both node and graph classification settings show encouraging results.

III. Preliminaries

Let $G = (V, E, X)$ denote a graph with a set of nodes V, edges E and a node feature matrix $X \in \mathbb{R}^{n \times d}$, where n is the total number of nodes and d is the dimension of the feature vector associated with each node. We represent the feature vector associated with the node, v, by x_v. For some positive integer k, let $N_v(k)$ be the set of nodes in the k-neighborhood of node v, i.e., nodes that are at most distance k from v. $N_v(k)$ also includes v. For a fixed k, let s_v^k, or s_v when k is clear from context, be an induced subgraph of G on $N_v(k)$. Let $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$ be the family of all such subgraphs. We refer to s_v as a *k-order subgraph* at node v. Let L indicates the Laplacian matrix of the graph and Φ is the matrix consisting

of normalized and mutually orthogonal eigenvectors of L. Let Λ represents the diagonal matrix of the eigenvalues of the Laplacian. We define a subgraph embedding as a function ϕ that extracts a compact and expressive signature of a k-order subgraph, $\phi : \mathcal{G} \rightarrow \mathbb{R}^{d'}$, where $\mathcal{G}$ is the family of all finite graphs and d' is the required dimension of latent feature space, which may be different from d.

IV. Ego-centric Spectral subGraph Embedding Augmentation Framework

In this section, we outline the details of our Ego-centric Spectral subGraph Embedding Augmentation (ESGEA) framework. The proposed scheme is divided into four phases: (1) ego-centric subgraph extraction around each node, (2) subgraph embedding design (ϕ), (3) feature augmentation and, (4) learning module. In the forthcoming sections, we outline the details of these phases one by one.

A. Ego-Centric Subgraph Extraction

Locally induced subgraphs capture the topological information of nearby nodes, enabling similar representations for nodes with identical subgraphs. They encode distinct attributes that are not always determined by explicit node features or properties based on the overall topology of the network. Subgraphs feature non-trivial internal structure, border connectivity, and concepts of neighborhood and position in relation to the remainder of the graph. They are, therefore, conducive to learning effective graph representations.

The importance of subgraphs can be further motivated by the famous reconstructibility conjecture, which holds true for many graph families, including but not limited to regular graphs, trees, Eulerian graphs, outer planner graphs, and graphs with at most 9 vertices [1], [2], [35]. The conjecture states the following:

Conjecture [2], [19], [38] *A graph with at least three vertices can be constructed uniquely (up to isomorphism) from a collection of its vertex-deleted subgraphs.*

In other words, if s_v is a subgraph of $G = (V, E)$ obtained by deleting vertex v and its incident edges, then s has a unique (up to isomorphism) collection of vertex deleted subgraphs $\{s_1, s_2, \cdots, s_n\}$. Variants of this conjecture deal with different subgraphs of s (e.g., [22], [23]). The primary assertion here is that *the topological and structural properties of a graph can ensue from its subgraphs.*

This discussion inspires and motivates the study of subgraphs for graph learning. For our purpose, we generate a k-order subgraph for each node v, which is essentially a subgraph induced on the nodes that are at most distance k from node v. Subsequently, we generate an embedding for each node v based on its k-order subgraph (as discussed in the next subsection). Depending on a node's structural role or position, the corresponding k-order subgraph may have different structural features. For instance, as shown in the simplest example k=1 in Figure 2, 1-order subgraphs of leaf nodes (B, E, F, and H) are the same (up to isomorphism). We note that non-leaf nodes generate 1-order subgraphs distinct from leaf nodes and as k increases more nodes are likely to have distinct embeddings.

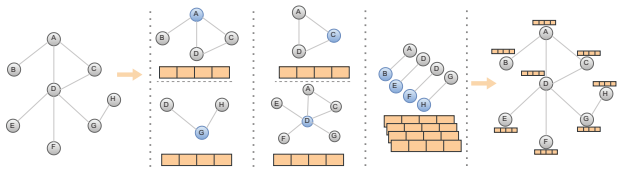

Figure 2: An illustration of constructing the simplest ego-centric subgraph embeddings (with 1-order subgraphs). For each node (indicated in blue), an induced subgraph from its immediate 1-order neighborhood is extracted and embeddings obtained. The obtained embeddings are finally associated with each node, respectively.

This subgraph embedding approach improves the learning of GNNs, particularly when abnormal or no node features are provided. Moreover, it enables a fairly broad approach by bridging the gap between GNNs and graph descriptors [31], [37], [40].

B. Design of Subgraph Embeddings

Designing a subgraph embedding that encode structural information at all scales, along with succinctness and expressivity, is a challenging task. Network Laplacian Spectral Descriptor (NetLSD) [37] was designed to satisfy most of the required properties that a graph descriptor should possess. For instance, the permutation invariance, extracting small, medium and large scale information, and time and memory efficient. NetLSD is based on the idea of diffusion in graphs, for example, how the heat diffuses across the nodes in the graph. The heat diffusion process over the graph at time t is examined through the *heat kernel* H_t, which is the matrix exponential $H_t = e^{-Lt}$, where L is the graph Laplacian and t is the time. Since L can be factorized as, $L = \Phi\Lambda\Phi^\top$, where Φ is the matrix consisting of normalized and mutually orthogonal eigenvectors, and Λ is a diagonal matrix consisting of the eigenvalues of L, we obtain the following:

$$H_t = e^{-Lt_i} = e^{\Phi(-\Lambda t_i)\Phi^\top} = \Phi e^{-\Lambda t_i}\Phi^\top. \tag{1}$$

where the ij^{th} entry of H_t indicates the amount of heat transferred from node v_i to v_j at time t_i. Similarly, the *wave kernel*, on the other hand, measures the propagation of mechanical waves across the graph. NetLSD is then defined by the traces of the heat or wave kernels at various time intervals: $h_{t_i} = tr(H_{t_i})$, which allows extracting more global information over time. In addition, they are permutation- and size-invariant and scale-adaptive. In terms of time complexity, the Laplacian spectrum requires $O(n^3)$ time and $O(n^2)$ memory, which hinders its scalability. Thus, the authors opted for block Krylov-Schur implementation in SLEPc [16], [32] to compute only λ extreme eigenvalues, thereby reducing the computation time and making NetLSD a fitting choice for the subgraph embedding.

C. Feature Aggregation with ESGEA

Aggregation functions play a crucial role in the representation and modeling of graph-structured data, specifically in the message passing framework, and has received significant attention

in the literature [15]. The performance and representational power of the models are significantly influenced by the selection of aggregation functions. For instance, several studies show that *sum* aggregation allows learning of graph structural properties [43]. Likewise, the *mean* aggregation is often employed to capture the distribution of the elements under consideration, while the *max* aggregation is commonly utilized to identify the most representative elements [43]. Given x_v and $\phi(s_v)$, we define a general aggregation function as follow:

$$x'_v = f(x_v, \phi(s_v))$$

x_v and $\phi(s_v)$ correspond to the node feature vector and subgraph embeddings, respectively. The function $f(.)$ can be substituted with aggregation such as *mean, max, min* or *sum*, depending upon the choice of the method. Because simple aggregations like *mean, max* and *sum* may result in the loss of valuable information, several recent studies have suggested using multiple aggregations, feature concatenation [15], aggregation in hierarchical fashion, as well as learnable aggregations [24]. Learnable aggregations typically entail the utilization of different techniques such as multi-layered perceptron or deep neural networks, which are capable of encoding intricate details and nuances of the input representations. Such methods have demonstrated significant potential in achieving superior performance in a wide range of applications, and are thus the subject of extensive research and investigation in the field.

In our proposed framework, we offer an adaptable aggregation module that can integrate any of the current aggregation operators to amalgamate the embeddings obtained from the subgraph to the primary node embeddings. However, it is crucial to take into account the limitations of these operators before deciding on the most suitable aggregation approach. While the use of learnable aggregations has its advantages, it is important to note that these methods can be computationally complex, posing challenges in terms of scalability and efficiency. Additionally, the traditional *mean, max*, and *sum* aggregation operators may not always be appropriate, especially in cases where the dimensions of the embeddings (d' and d) are unequal. In such cases, a simple combine function such as feature concatenation is a viable alternative, which can work effectively in all scenarios and potentially yield improved results. Figure 3 illustrates our methodology for extracting subgraph, computing spectral embeddings, and feature augmentation. The illustration shows a graph with three corrupted (in gray) and four complete node features (in green) and highlights the overall subgraph feature extraction and augmentation approach.

D. Graph Learning Module for ESGEA

This section focuses on *Message Passing Graph Neural Networks (MPNNs)*, which uses an iterative learning approach to acquire graph representations. MPNNs retain a representation vector $h_v^l \in \mathbf{R}^{\hat{d}}$ for each node $v \in V$ in a given a graph $G = (V, E, X)$. Note that $\hat{d}$ is the size of the embedding h, which may vary from d and d' which are the sizes of the input

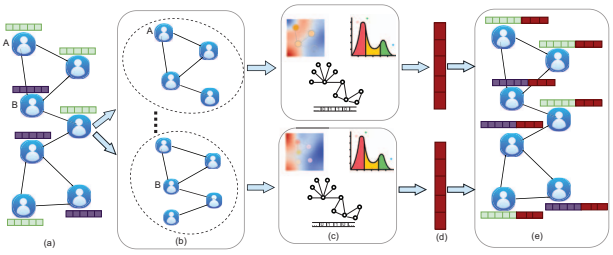

Figure 3: Illustration of the overall ego-centric subgraph extraction, embeddings and aggregation: (a) is the input graph with normal (green) and abnormal (gray) node features; (b) demonstrates step 1, i.e., the generation of $k-$order subgraphs for each node; (c) shows the use a set of graph descriptors for extracting graph embeddings; (d) indicates the corresponding embeddings extracted for each subgraph; and (e) illustrates feature aggregation (concatenation in this case) of subgraphs feature vectors with their original features.

Algorithm 1: ESGEA for Graph Classification

Input: Graph $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$, subgraph depth: k, #Layers: L, non-linearity: σ Weight matrices W^l
Output: vector output z_g for all $g \in \mathcal{G}$

for G *in* $\mathcal{G}$ **do**
  for *node* v *in* G **do**
    $s_v \leftarrow Extract\_subgraph(G, v, k)$;
    $h'_v \leftarrow \phi(s_v)$
    $x'_v \leftarrow h'_v$
  $h_v^{(0)} \leftarrow x'_v \; \forall v \in G$
for G *in* $\mathcal{G}$ **do**
  for $l = 1, \ldots, L$ **do**
    for *node* v *in* G **do**
      $a_v^{(l)} = f_{\text{AGG}}\left(h_u^{(l-1)} \mid u \in \mathcal{N}(v)\right)$
      $h_v^{(l)} = f_{UPDATE}^{(l)}\left(h_v^{(l-1)}, a_v^{(l)}\right)$
    $h_g^{(l)} \leftarrow h_v^L \; \forall v \in G$
return $z_g \leftarrow h_g \; \forall g \in \mathcal{G}$

features and subgraph embeddings respectively. MPNNs are defined as follows:

$$h_v^{(0)} = x_v \; \forall v \in V \tag{2}$$

$$a_v^{(l)} = f_{AGG}^{(l)}\left(h_u^{(l-1)} | u \in \mathcal{N}(v)\right) \tag{3}$$

$$h_v^{(l)} = f_{UPDATE}^{(l)}\left(h_v^{(l-1)}, a_v^{(l)}\right) \tag{4}$$

Node features $h_v^{(0)}$ are initialized with the original node features x_v and then the aggregate and update functions are used to update the node features based on its neighbors, and the prior state at every iteration.

In Algorithm 1, we provide a step by step procedure of the proposed framework in a graph classification setting. The algorithm first extracts subgraph embeddings for each node using an input parameter k, which is the depth of the subgraphs. These subgraphs are then supplied to the embedding function to produce graph embeddings. We would like to note that we propose spectral graph embeddings, e.g., NetLSD [37] for generating subgraph embeddings which distinguishes our work from the existing nested subgraphs GNN methods. As described

in section IV-B, spectral descriptors are powerful methods for extracting expressive graph embeddings. Nonetheless, our proposed framework is general, thus any embedding method may be used in this step. As we do not consider node features in social networks, we assign subgraph embeddings h'_v as node features in the graph classification setting (step 5 and 8). In addition, we provide a generalized MPNNs strategy for learning representations for each graph that may be used for the subsequent ML task. $f_{AGG}(.)$ and $f_{UPDATE}(.)$ are generic functions that may be fine-tuned based on the learning architecture of choice.

Similar to the graph classification, node classification is a well-studied problem, particularly in semi-supervised learning. It has numerous applications in several fields, such as online social networks, biological networks, and ecommerce networks. Node classification involves training a model in a supervised or semi-supervised setting that can predict a label for a new unseen node. In a GNN setting, we obtain node representations from the last layer followed by a linear layer to obtain the class label. A loss function is then applied to train the model accordingly. We define the cross entropy loss function as follows that we use for binary classification.

$$\mathcal{L} = -\frac{1}{N}\sum_{j=v}^{N} y_j \, log(p_j) + (1 - y_j) \, log(1 - p_j) \qquad (5)$$

where p_i is the Softmax probability obtained for the data point j, and y_j is the corresponding ground truth value.

To adapt Algorithm 1 to the node classification task, a few modifications can be made. Specifically, the subgraph embeddings generated by the embedding function (step 4) are integrated with the node features using an aggregation function, which is thoroughly explained in IV-C. The resulting aggregated node features are then fed into the learning framework to obtain node representations. In contrast to the graph classification setting, where we process a collection of graphs (step 1 and 7), in this case we iterate solely over the nodes of a single graph and learn node embeddings. Thus, the proposed learning framework provides a flexible MPNNs-based solution for node classification as well.

The Algorithm 1 introduces a novel framework that primarily involves two crucial input parameters impacting the model's performance: (a) the depth of subgraphs (k) and the number of GNN layers L. The combination of these parameters plays a crucial role in feature aggregation and the receptive fields of the models. Figure 4 demonstrates the comparative merits of different combinations of these parameters. A larger value of k and L for a given training node can increase the overlap of information obtained from the node's neighborhood. In addition, it increases the model's receptive field and may result in oversmoothing that hinders the model's performance. Similarly, a combination of large and small input values widens the distance between the embeddings and message passing, which may result in a reduction in performance. Conversely, combining both parameters can result in superior performance.

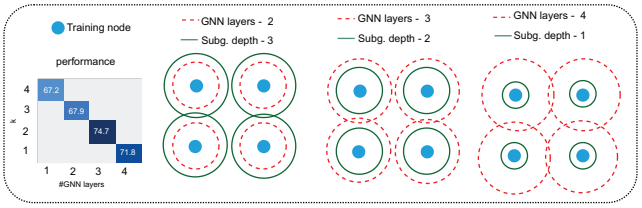

Figure 4: An illustration of the comparative merits of GNNs and subgraph depth with respect to their receptive fields within the graph and resultant performance reported on LastFM Asia dataset.

V. Experimental Evaluation

To assess the performance of the proposed framework, here we define the following two research questions.

- **RQ-1** Can the proposed ESGEA framework improve performance in the absence of explicit node features?
- **RQ-2** Does the effectiveness of the proposed ESGEA framework remain intact to improve performance in applications where node features are unreliable?

To answer the first question, we consider a graph classification setting in which node features are unavailable. For the second research question, we consider node classification setting with abnormal node features. The forthcoming sections detail the experimental design and results.

General Setup: We ran all the experiments on a 112-core Intel Xeon CPU 2.20 GHz machine with 512 GB of RAM and an Nvidia GPU with 48 GB of memory. Each method is trained on 80% and tested on 20% of the dataset. We run all methods with 10 different seeds, and average accuracy and Area Under the Curve (AUC) are reported for node classification and graph classification tasks, respectively. Throughout the experiments, we consider the depth of the subgraphs to be 3 and use NetLSD descriptor to construct subgraph embeddings. The source code is made publicly available[1].

A. Graph Classification

When node features are unavailable, current graph classification techniques typically rely on one-hot-degree encoding to provide relevant node features for GNNs in order to improve their performance. However, in the majority of cases, these features do not provide a stronger learning basis for GNNs. Here, we hypothesize that the features generated through the proposed approach improve the performance of models for datasets without node features. We test our hypothesis in the graph classification setting on five datasets where node features are unavailable.

Datasets: We consider *Github Stargazers*, *Reddit threads*, *Reddit Binary*, *Deezer Egos*, and *Twitch Egos* social network datasets in our experimental setup. Due to space limitation, we refer the reader to [28] for further dataset details.

Baselines: We consider the following backbone GNN models, including which include the seminal GCN, GraphSAGE, and Residual Gated GCN (RGGCN), along with more recent advanced model UniMP and provably more expressive k-GNN.

[1]ESGEA publicly available code: https://github.com/Anwar-Said/ESGEA

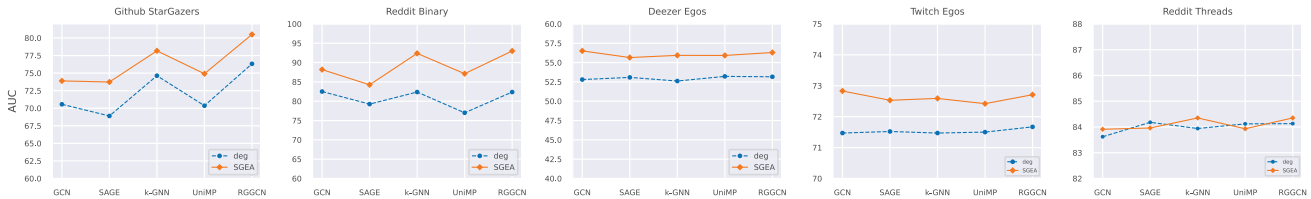

Figure 5: Comparison of mean AUC scores (10 runs with different random seeds) against *one-hot-degree-encoding* (*deg*) as node features on five different GNN models in graph classification setting.

Table I: Comparison of node classification accuracy on the test set of Facebook dataset across 10 seeded runs[1].

Model	Corruption ratio					
	0%	10%	20%	50%	80%	90%
MLP	76.30	67.96	61.48	43.29	32.84	30.70
ESGEA	76.81	68.49	62.10	43.51	31.14	30.56
GCN	93.88	92.01	89.89	83.60	77.08	75.60
ESGEA	93.89	92.08	90.16	83.67	78.23	76.75
GAT	93.86	85.06	72.80	59.01	54.85	59.82
ESGEA	94.01	85.96	81.09	67.77	66.19	63.03
AirGNN	90.01	88.06	86.27	79.53	64.80	61.46
ESGEA	90.02	87.80	86.14	79.54	66.15	58.61
UniMP	95.16	94.44	93.05	85.03	78.44	77.80
ESGEA	95.27	94.31	93.22	89.41	85.89	85.13

Experimental setup: Initially we generate node features for each node with subgraphs embedding using NetLSD descriptor. The dimension of NetLSD descriptor is set to 20 and depth of the subgraphs k is set to 2 (small diameters) and 3. We then run each of these models with the generated NetLSD embeddings and one-hot-degree encoding and compare the results. Each model consists of three convolution layers followed by SortPooling layers [45], two 1D convolutions and three MLP layers. The train:test splits ratio was set to $80:20$, batch size to 128, learning rate to $1e^{-4}$ and the number of epochs was set to 100. We consider Area Under the Curve (AUC) similar to [28] as the evaluation metric.

We report the performance comparison in terms of AUC with degree encoding as node features in Figure 5. The results demonstrate that ESGEA vastly outperforms one-hot-degree encoding. More specifically, Residual Gated GCN and k-GNN demonstrate an encouraging improvement of up to 10% on the Reddit binary dataset. Similarly, each method has received up to 4% improvement on the Github StarGazers dataset. ESGEA also outperforms throughout on Twitch Egos and Deezer Egos datasets. These results clearly illustrate the effectiveness of the proposed approach in the applications where node features are unavailable.

B. Node Classification

In the node classification setting, we evaluate the proposed method in a setting where a certain percentage of node features is corrupted. We apply the proposed approach to enrich the corrupted node features and then trained different GNNs to evaluate the results. In the following sections, we describe the datasets, experimental setup, and results in detail.

[1]The results are colored red where the proposed method ESGEA outperforms the baseline and are colored blue otherwise

Table II: Comparison of node classification accuracy on the test set of LastFM Asia dataset across 10 seeded runs[1].

Model	Corruption ratio					
	0%	10%	20%	50%	80%	90%
MLP	71.76	59.50	49.43	31.13	21.62	20.24
ESGEA	71.85	60.00	50.39	30.32	20.98	20.25
GCN	86.11	84.28	82.36	74.94	67.50	66.03
ESGEA	86.50	84.55	86.47	75.43	68.04	66.89
GAT	85.52	79.37	78.03	72.20	67.65	70.91
ESGEA	85.72	79.52	85.72	75.42	73.21	75.24
AirGNN	86.45	85.76	85.69	82.31	79.37	77.17
ESGEA	86.39	85.70	85.70	83.35	78.50	77.36
UniMP	87.12	85.69	82.98	77.47	72.07	71.70
ESGEA	87.28	84.58	87.20	80.61	78.03	78.54

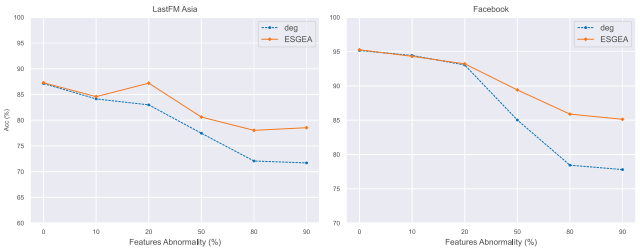

Figure 6: ESGEA with UniMP performance on both Facebook and LastFM Asia datasets. Here we vary the percent of abnormality injected into the original node features. The results are compared with the standard one-hot-degree encoding (denoted as deg).

Datasets: We consider two social network datasets, *Facebook Page-page* and *LastFM Asia*, for the node classification task. We refer the reader to [27] for further details on these datasets.

Backbone GNNs: In our evaluations, we consider four baseline methods: a two-layered MLP, GCN [20], GAT [39], Unified Message Passing (UniMP) [34], and AirGNN [25].

Setup: We examine a three-layered architecture for every method implemented in PyTorch Geometric. The number of hidden channels was set to 16, learning rate as 0.01, k and L equal 3 and the weight decay was set to $1e^{-4}$. Each model was trained for 200 epochs with 50 epochs as an early stopping criterion. Throughout the experiments, we evaluate all models on $0\%, 10\%, 20\%, 50\%, 80\%$, and 90% noise setting, i.e., the number represents the percentage of nodes for which original node features were replaced with random features sampled from a Gaussian distribution as done in [25].

Results: Table I and II present the classification results of our evaluation on both datasets. The proposed framework achieves either comparable or superior performance in every

experiment. Specifically, the proposed method with UniMP and GAT models boasts a performance gain of up to 7% on the Facebook and LastFM Asia datasets. In Figure 6, we visualize the performance of ESGEA in conjunction with UniMP on both Facebook and LastFM Asia datasets. We observe as the abnormality increases, ESGEA is less impacted as degree encoding and maintains stronger performance.

C. Parameters Sensitivity and Runtime

Graph neural networks that employ subgraph representations are a novel and sophisticated category of expressive learning techniques designed to represent graphs as an amalgamation of subgraphs [13]. The efficacy of these methods is typically contingent upon the depth of the subgraphs as well as the number of GNN layers utilized, both of which constitute hyper-parameters. As the number of layers in MPNNs increases, a common phenomenon known as oversmoothing occurs, whereby the node features begin to converge into indistinguishable vectors. This is evidenced in Figure 1, which shows the performance of GCN on the Facebook dataset. Conversely, while a deeper subgraph results in an enlarged receptive field, it also exacerbates the oversmoothing effect, as node features are smoothed too quickly. Moreover, it also increases the running times of the methods by several folds. It is therefore imperative to make informed selections for these hyper-parameters in order to facilitate the training of the model. To delve deeper into the interplay of these parameters, we conducted an empirical analysis on both the node and graph classification tasks.

Graph Classification: We use Reddit threads dataset to analyze $k-$GNN with varying numbers of layers and subgraph depths, as illustrated in Figure 7 (a). Our objective was to determine how these parameters affect performance. We observe that a smaller or larger number of GNN layers reduces the model's performance, whereas a layer count of 3 or 4 yields optimal results. We also assessed the execution time of subgraph embedding and model training on the Deezer egos and Reddit threads datasets, as displayed in Figure 7 (b). These datasets have small diameters averaging at 3.4 and 4.5. As a consequence of their low diameters, the computation time for subgraph embedding is not extensive and can be completed within 50 seconds. Similarly, model training times fall within the range of $200 - 300$ seconds. These findings demonstrate

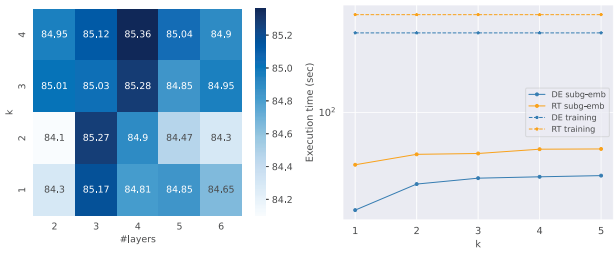

Figure 7: Graph classification parameter sensitivity and runtime analysis: (a) Performance comparison of $k-$GNN on Reddit threads dataset. (b) running times of subgraph embeddings and training times of GCN on Reddit threads and Deezer egos datasets.

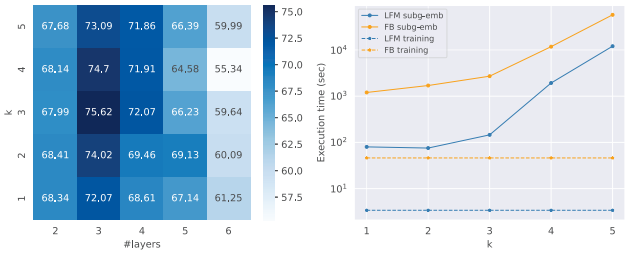

Figure 8: Node classification parameter sensitivity and runtime analysis: (a) Performance comparison of Graph attention network on LastFM Asia dataset. (b) running times of subgraph embeddings and training times of GCN on LastFM Asia and Facebook datasets.

that computing embeddings using subgraphs and NetLSD descriptors is scalable on large graphs, without sacrificing the method's efficiency.

Node Classification: To assess the effectiveness of the hyper-parameters and analyze the running times, we consider both Last FM Asia and Facebook social networks in a node classification setting. As illustrated in Figure 8 (a), our results indicate increasing both the number of layers and the depth of subgraphs can lead to suboptimal performance. Our analysis suggests a favorable combination of hyper-parameters involves three GNN layers with depth $k = 3$ for optimal results. In Figure 8 (b), we present the running time of the NetLSD descriptor for computing subgraphs of varying depth on the LastFM Asia and Facebook datasets. We also compare the training times of GCN on both datasets. Notably, as the depth of the subgraphs increases, their size grows, which consequently extends the running time of the descriptor as depicted in Figure 8 (b). Nonetheless, we observe that subgraph embeddings for $k = 3$ were calculated within a reasonable 2 and 45 minutes accordingly.

VI. Conclusion

In this paper, we proposed Ego-centric Spectral subGraph Embedding Augmentation (ESGEA), a novel framework for extracting node features from network topology, which is especially important for settings where networks have missing or unreliable node feature information. Preceding message passing, our framework leverages the node's neighborhood topological structure from spectral graph embeddings obtained on extracted ego-centric k-order subgraphs to generate informative and expressive node features, which are then augmented to the feature matrix using a suitable aggregation approach. Notably, our framework is flexible and compatible with any GNN-based architecture, and exhibits impressive performance. Our detailed evaluations on seven datasets, and eight baselines, encompassing both node and graph classification settings, illustrate the efficacy and potential of the proposed approach.

Acknowledgment

This material is based upon work supported by the National Science Foundation Grant Nos. 2239881, 2325416, and 2325417.

REFERENCES

[1] Mark Bilinski, Young Soo Kwon, and Xingxing Yu. On the reconstruction of planar graphs. *Journal of Combinatorial Theory, Series B*, 2007.

[2] J Adrian Bondy and Robert L Hemminger. Graph reconstruction—a survey. *Journal of Graph Theory*, 1(3):227–268, 1977.

[3] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[4] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

[5] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *NeurIPS*, 33, 2020.

[6] Nicholas Choma, Federico Monti, Lisa Gerhardt, Tomasz Palczewski, Zahra Ronaghi, Prabhat Prabhat, Wahid Bhimji, Michael M Bronstein, Spencer R Klein, and Joan Bruna. Graph neural networks for icecube signal classification. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 386–391. IEEE, 2018.

[7] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[9] Tyler Derr, Yao Ma, Wenqi Fan, Xiaorui Liu, Charu Aggarwal, and Jiliang Tang. Epidemic graph convolutional network. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 160–168, 2020.

[10] Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018.

[11] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th CIKM*, 2021.

[12] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[13] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 2017.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[16] Vicente Hernandez, Jose E Roman, and Vicente Vidal. Slepc: A scalable and flexible toolkit for the solution of eigenvalue problems. *(TOMS)*, 31(3):351–362, 2005.

[17] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

[18] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

[19] Paul Joseph Kelly. *On isometric transformations*. PhD thesis, University of Wisconsin, 1942.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Alexandr V Kostochka and Douglas B West. On reconstruction of graphs from the multiset of subgraphs obtained by deleting ℓ vertices. *IEEE Transactions on Information Theory*, 2020.

[22] Vladimir Levenshtein, Elena Konstantinova, Eugene Konstantinov, and Sergey Molodtsov. Reconstruction of a graph from 2-vicinities of its vertices. *Discrete Applied Mathematics*, 2008.

[23] Vladimir I Levenshtein. A conjecture on the reconstruction of graphs from metric balls of their vertices. *Discrete Mathematics*, 2008.

[24] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv arXiv:2006.07739*, 2020.

[25] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. Graph neural networks with adaptive residual. *NeurIPS*, 34:9720–9733, 2021.

[26] Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. *arXiv preprint arXiv:2111.12128*, 2021.

[27] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[28] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM (CIKM '20)*. ACM, 2020.

[29] Anwar Said, Roza G Bayrak, Tyler Derr, Mudassir Shabbir, Daniel Moyer, Catie Chang, and Xenofon Koutsoukos. Neurograph: Benchmarks for graph machine learning in brain connectomics. *arXiv preprint arXiv:2306.06202*, 2023.

[30] Anwar Said, Saeed-Ul Hassan, Waseem Abbas, and Mudassir Shabbir. Netki: a kirchhoff index based statistical graph embedding in nearly linear time. *Neurocomputing*, 433:108–118, 2021.

[31] Anwar Said, Saeed-Ul Hassan, Suppawong Tuarob, Raheel Nawaz, and Mudassir Shabbir. Dgsd: Distributed graph representation via graph statistical properties. *Future Generation Computer Systems*, 2021.

[32] Anwar Said, Mudassir Shabbir, Saeed-Ul Hassan, Zohair Raza Hassan, Ammar Ahmed, and Xenofon Koutsoukos. On augmenting topological graph representations for attributed graphs. *Applied Soft Computing*, 136:110104, 2023.

[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[34] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[35] Hannah Spinoza and Douglas B West. Reconstruction from the deck of-vertex induced subgraphs. *Journal of Graph Theory*, 2019.

[36] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[37] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2347–2356, 2018.

[38] Stanislaw M Ulam. A collection of mathematical problems. *New York*, 29, 1960.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[40] Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

[41] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020.

[42] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv arXiv:1810.00826*, 2018.

[44] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[45] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[46] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1399–1407, 2019.

[47] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *ACM SIGKDD*, 2018.