FEGR: Feature Enhanced Graph Representation Method for Graph Classification

Mohamad Abushofa
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
m.e.a.abushofa2@ncl.ac.uk

Amir Atapour-Abarghouei
Department of Computer Science
Durham University
Durham, United Kingdom
amir.atapour-abarghouei@durham.ac.uk

Matthew Forshaw
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
matthew.forshaw@ncl.ac.uk

A. Stephen McGough
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
stephen.mcgough@newcastle.ac.uk

***Abstract*—Graph representation plays a key role in graph analytics to perform a variety of downstream machine-learning tasks. This paper presents a novel method for extracting expressive graph representation based on a combination of statistics captured from a graph and node properties. We use both local and global-level information along with the original node properties to extract a meaningful feature representation of the graph. This allows us to build expressive graph descriptors that can be run with limited training data and computational resources and achieve competitive results. We discuss the merits of the proposed approach in terms of sensitivity, running times, and stability. Our evaluation of various graph classification benchmark datasets shows that the proposed method either outperforms or provides similar results to state-of-the-art methods. We further outline the potential future directions in graph machine learning research.**

***Index Terms*—Graph representation; concatenated features; Non-parametric approach; Graph descriptor.**

I. Introduction

Graph representation has a long and illustrious history of being used to solve complicated real-world problems covered in many fields, such as chemistry, biology, and computer science. A graph is formed from a collection of nodes which are connected together via edges. Both nodes and edges may be tagged up with data – called here properties (e.g., in a social network, nodes representing people may have properties of name and age). A range of approaches to studying network activity and solving real-world problems such as node classification, graph classification, link prediction, and community discovery, have recently been developed [1]. Machine learning algorithms have had great success in the last decade on a range of applications, such as computer vision, image classification, and speech generation. However, these can not be directly applied to graph structured data as such methods require data to be in structured tensor spaces, while graphs have complex and combinatorial structures. Thus, an expressive representation of graph structure in a structured (vector) space is required to apply machine learning methods to graph problems.

One of the more recent areas of research is graph representation, which tries to represent a graph in a low-dimensional Euclidean space [2]. The basic premise is to represent a graph (containing real-world entities and relationships) as points in a low-dimensional feature space whose geometry shows the correspondence between the entities – thus embedding the graph within a vector space. The graph embedding problem is closely connected to traditional machine learning data compression tasks, such as multi-dimensional scaling and dimensionality reduction [3], in which large amounts of data are reduced or compressed to lower dimensional space. To find more tractable representations, these approaches look for repeats and regularities in the data. Graph embedding methods, on the other hand, necessitate taking into account the full latent geometry and must meet permutation invariance (i.e., yield the same representations for isomorphic graphs), scalability (manage graphs of varying sizes), memory efficiency, expressiveness (good embedding or representation), and computational efficiency requirements. Meeting all of these requirements at the same time makes graph representation challenging [4].

Graph representation approaches have seen considerable uptake in recent years, with a number of approaches proposed. These approaches are divided into two categories: Graph Neural Networks (GNNs) [5] and graph descriptors. GNNs use an end-to-end learning architecture to learn the desired embeddings by utilising node features. On the other hand, graph descriptors are typically based on graph spectrum (set

ASONAM '23, November 6–9, 2023, Kusadasi, Turkey © 2023 Copyright is held by the owner/author(s). ACM ISBN 979-8-4007-0409-3/23/11. https://doi.org/10.1145/3625007.3627600

of graph eigenvalues of the adjacency matrix) [6] and other metrics and do not leverage node features, resulting in a significant false positive rate [7], [8]. There have been attempts to use graph descriptors such as using a 'fingerprint' of metrics to represent a graph [9] and Distributed Graph Statistical Distances (DSGD) [1] that extracts graph representation in centralised and distributed environments using very simple graph statistics. However, to the best of our knowledge, there is no prior work which uses only features which are concatenated from real properties from the graph with extracted features from the graph and node metrics. We hypothesise that this may increase the performance of graph machine-learning tasks.

We propose a novel graph descriptor we denote FEGR (Feature Enhanced Graph Representation). FEGR takes advantage of statistical measures used in graph fingerprints [9], while simultaneously addressing their limitations by adding node properties to the statistical measures to produce expressive graph embeddings. We focus on the experimental setting of DGSD [1] to compare. In this study, we use two types of features: local and global features to construct the graph signature. For the local features, we consider the degree, clustering coefficient, average degree of neighbours, and average clustering coefficient of neighbours. Aggregating these local features is not enough to produce a good representation of the structure of the entire graph. Thus, for the global features, we consider the largest five eigenvalues, the number of connected components, the total number of nodes, and the total number of edges in the graph. We initially, compute the local features and then take the median, mean, standard deviation, skewness, and kurtosis to form the graph descriptor. We finally compute the global measures and concatenate these with the local features and features extracted from the properties of nodes/edges. Our main contributions are thus summarised as follows:

- We enrich the graph embedding by incorporating aggregated original feature information associated with each node with local and global graph measures.
- We explore the graph representation problem through various graph theoretical tools such as `networkx` and `graph-tools` and subsequently utilise nine features for bioinformatics graph classification.
- We evaluate the proposed method in different experimental settings, including graph classification, sensitivity, and the evolution of nodes and edges in a graph.
- We have made the source code publicly available to enable better reproducibility of the results:[1].

II. Motivation and Requirements

We provide the motivation behind this work by depicting an example of a graph machine learning (ML) application. Through this example, we depict a number of issues facing graph representation, such as accuracy and scalability. Accordingly, we elaborate on how these issues can be mitigated. Before, going to the motivation scenario, we discuss graph machine learning tasks as follows:

[1]https://github.com/mohamad-1977/FEGR

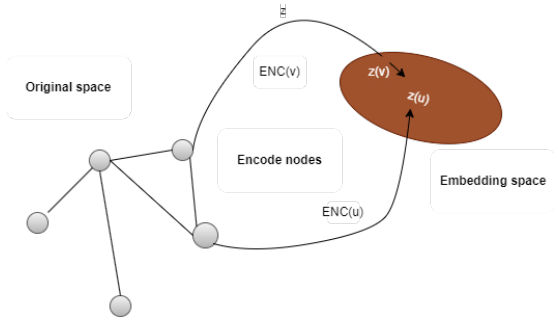


Fig. 1: Graph Representation Framework

A. Graph Representation

Formally, the graph representation problem can be defined as follows. Given a graph $G = (V, E)$, where V is the set of vertices (sometimes called nodes) and E is the set of edges (sometimes called links), the main goal is to design a function $f : G \rightarrow \mathbb{R}^d$, where $d << |V|$ which encodes or embeds the node v, $ENC(v)$, to produce $z_v \in R^d$, As an example, the embedding of node v in the embedding space is shown in Figure 1. The graph representation problem is challenging due to the combinatorial nature of graphs. The representation function f should satisfy several properties including but not limited to: i) permutation invariance – that the function does not depend on the arbitrary ordering of the rows/columns in the adjacency matrix, ii) time and memory efficiency, iii) flexible embedding size, iv) stable results, and v) scalability.

Before, discussing the motivation scenario, we discuss common graph machine-learning tasks that can benefit from more robust graph representations as follows:

1) Graph Classification: Given a set of graphs $G = g_1, g_2, \ldots, g_n$ with their corresponding labels $Y = y_1, y_2, \ldots, y_n$, the goal is to learn/extract a representation h_g for the given graph to predict a label y_g for an unseen graph g. The graph classification problem has diverse applications in various domains, such as bioinformatics, social networks, communication networks, and computational biology [10]. For example, in bioinformatics, we can perform toxicity prediction using graph classification where molecular graphs with their corresponding labels (binary, in this case, toxic and not toxic) can be provided as input to train a graph classification model. This can be used to predict the label of a new molecular graph [11].

2) Link Prediction: In the link prediction task, the goal is to learn a representation vector h_v for each node v to predict a link between two nodes v_i, v_j, which is a function of $f(h_i, h_j)$. Link prediction is widely adopted for knowledge graph completion and recommendation systems in diverse applications.

B. Motivation Scenario

Recently, graph machine learning applications have found their way into the healthcare sector [12]. Some medical data are represented as graphs, such as patient records and diseases [13]. Therefore, this data requires a model to convert it to

the embedding space. This representation then goes on downstream machine learning tasks, such as graph classification, node classification, or link prediction. Thus, finding a robust representation model leads to better downstream results. In contrast, a low representation model leads to low accuracy, which may harm decisions required in the health sector. However, graph representations can give rise to some issues as well as challenges. For instance, as the computational complexity of graph representation approaches often grows exponentially with the number of nodes, these approaches are often unsuitable for massive graphs. Therefore, it is crucial to identify the existing issues and challenges, and herein lies the need for a robust model for this representation. As such, our approach is driven by the following requirements:

- Scalability: the new approach should be highly scalable, to graphs of millions of vertices / edges, and capable of doing the classification in a reasonable time. Ideally, the approach should be portable to a many-core or distributed graph processing system to help with scalability.
- Sensitivity To Graph Size: our approaches should take the size and order of the graphs into consideration.
- Sensitivity To Topologies: it should be able to detect the difference between graphs which are highly structurally and topologically similar.

III. Related Work

Node features play a vital role in learning graphs [2]. Graph representation approaches have exploded in popularity in recent years, with a slew of new approaches being suggested. Graph Neural Networks (GNNs) [2] use an end-to-end learning architecture to learn the desired embeddings by utilizing node features. Graph descriptors, on the other hand, extract node embeddings using statistical measurements rather than exploiting node attributes. The main drawback of GNNs is that they require lots of training data, are computationally expensive, and thus they do not scale well to massive networks. Graph descriptors, on the other hand, are typically based on graph spectrum and other metrics and do not leverage node properties, resulting in a large false-positive rate. To overcome these drawbacks, we propose a novel graph descriptor we denoted as FEGR (Feature Enhanced Graph Representation).

We propose using eight aggregated graph-theoretic properties along with the aggregated node features to encode the graph into a lower dimensional space to deploy into any downstream machine learning task. In this section, we survey graph representation methods categorized into two categories: non-parametric and parametric approaches.

A. Non-parametric approaches

1) Direct Method: One graph can be converted to another graph by performing a series of edit operations. The nominal number of such edit operations is referred to as Graph Edit Distance (GED) [14]. Calculating and approximating GED has been shown to be NP-hard and APX-hard, respectively [15], [16]. This is because, the calculation and approximation entail defining the correspondence between the nodes of the graphs under comparison, which is always considered a difficult task [17]–[20]. More flexible distance definitions have been utilised in previous studies to improve on GED. These measures were based on the propagation models [21] or the vertex and path resemblances [22]. Despite these flexible measures being less sensitive to local alterations compared to GED, they require an advanced alignment of nodes, making them restricted to explicit applications. If this requirement is not met, the permutation matrix that defines the full node correspondence becomes relaxed, and a family of tractable distances (FTD) [23] is thus considered [24]. The relaxed matrix can easily be computed and is imperative in preserving the local properties; however, the resulting measure is still permutation-sensitive.

2) Kernel approach: The similarity functions among various graphs that operate by executing implicit alterations on the graph topologies when comparing two graphs are known as graph kernel [18]–[20]. An example of such similarity functions includes the shortest-path (SP) kernel [19] that counts the numbers of shortest paths that match with similar node labels at endpoints. Despite the graph kernel functions having specific valuable properties, they have not managed to attain both size-invariant and scale-adaptive comparisons of graphs. Also, the computational requirements of kernel functions make them inapplicable in large-scale comparisons of graphs. The Multi-scale Laplacian Graph kernel (MLG) [18] is capable of attaining scale-adaptivity by manipulating information propagation within the graph and adding up the information at every iteration. Nevertheless, this approach has also raised the concern of computational overhead being cubic in the eigenvalues of the Laplacian matrix. Another graph kernel approach called the Weisfeiler-Lehman can operate by refining the vertex colors [20]. The method operates by first assigning colours to the nodes using the vertices degrees, and then, in relation to the neighbour colours, enhances the vertex colours. The approach has been successful on benchmark datasets but remains insufficient when differentiating regular graphs. Another method called the deep graph kernel [25] utilizes the word embedding model and compares the number of motifs or subgraphs between the graphs under comparison. It is also an expensive approach since the motifs extraction is highly costly, especially when dealing with large networks, thus, making the method infeasible. Lastly, the shortest path kernel functions by encoding and comparing graphs using the shortest path between their vertices' pairs [26]. The random walk kernel functions by quantifying the graphs based on the number of shared walks among them [17].

3) Statistical Representations: Statistical properties can be used to develop a one-off graph signature vector that can be utilized in the comparison of graphs through representation-based methods. Properties like nodes and degrees have been used in preliminary works [27], since they focus on local characteristics, are oblivious to universal features, and are easy to compute. A more advanced representation known as the Family of Spectral Distances (FSGD) [28] has been applied to achieve a histogram representation on the dense biharmonic

graph kernel that is high-dimensional and sparse. Nonetheless, this advanced representation fails to capture important graph characteristics at dissimilar graph sizes or resolution scales, is complex in terms of quadratic time, and is thus impracticable for large graphs.

4) Spectral Representation: Spectral graph theory can be applied to effectively compare 3D objects [29]–[31]. Pairwise shape commonalities can be computed by discovering the nominal distortion embedding of one shape into another shape. Successful works have entailed filtering corresponding functions to diffusion models that are known, such as commute time distances, heat [32], [33], and wave [34]. However, it is unclear if the 3D filters can sustain expressiveness when dealing with high-dimensional graphs. The spectral graph theory provides a solid ground for comparing graphs.

Higher-order proximity is significant in embedding graphs into the vector spaces. Its successful functionality requires graph information from both the local and global levels. Thus, most graph embedding is achieved using the global-level and graph-theoretic measures. FGSD [27] produces a representation of a graph in the form of a histogram constructed from the pair-wise distances of the nodes that are computed using a graph spectrum that has both local and global information of the graph. In the same way, NetLSD [24] uses the heat diffusion process on graphs and generates a characteristic vector utilising the Laplacian spectrum. Some studies [35] have utilized simple statistical graph properties like clustering coefficient, average degree as well as central measures to encode graphs.

B. Parametric approaches

1) Graph Neural Networks: GNNs are powerful architectures for learning from graph-structured data [5]. They have recently received significant attention because of their promising results and their applications in the real world [36]. Various approaches have been proposed in recent years to extract graph representations in multiple settings [37]. Initial works in this area date back to that of Gori et al. [38] and Scarselli et al. [39]. These methods usually learn state vectors for all vertices in an end-to-end fashion. They use an information diffusion mechanism, defined by nodes updating their states and exchanging information by passing "messages" to neighbouring nodes until they reach a stable equilibrium to aggregate node states in their neighbourhood.

Li et al. [40] proposed a gated graph sequence neural network, a well-known GNN model that uses a Gated Recurrent Unit to learn the states of nodes. The network learns about the node and the previous states of the node neighbourhood to learn latent representations for each node. Similarly, Dai et al. [41] used a similar approach to node state learning but used a stochastic fixed point gradient descent instead of a Gated Recurrent Unit to accelerate the learning process.

2) Graph convolutions: Graph convolutions involve two basic steps of neighbourhood aggregation and pooling. The aggregate function involves a message-passing mechanism where node features are aggregated with their neighbours [42], whereas the pooling operation combines the embeddings to obtain graph-level representation [43]. Previous GCN studies were conducted by Kipf and Welling [44] and Hamilton et al. [45]. Kipf and Welling [44] aggregated the neighbourhood of a node while considering the entire adjacency matrix which requires the whole adjacency matrix and is thus transductive, whereas Hamilton et al. [45] used a message-passing mechanism to consider a subset of neighbours to aggregate, allowing inductive representation learning over graphs.

To focus on important or influential neighbours of each node in aggregation, Veličković et al. [46] presented an attention mechanism that learns attention parameters for each neighbour. More recently, Bonner et al. [47] proposed a TNA framework to learn dynamic network structure. Two TNA modules followed by two GCN layers were considered to learn the embeddings, then a variational inference was used to sample an embedding for each node, while the vector inner product was used as a decoder to predict the graph in the next time interval. Several follow-up approaches were subsequently introduced [48], [49].

IV. Methodology

A. Approach overview

Our approach builds on the ideas from Bonner *et al.* [9] in that we build a 'fingerprint' of a graph generated from a range of statistics captured from both local and global level features of the graph, though we use a different set of statistics here. We take this further by creating extra 'fingerprint' features derived from the properties of the nodes within the graph. In this way, we make a 'fingerprint' which captures both the topological nature of the graph but also the properties of the nodes.

B. Generating graph topological descriptors

1) Local features: To encode the local features of the graph, a number of vertex-level metrics are extracted. Although a wide selection of vertex feature metrics exist, each exhibits different characteristics in terms of topological sensitivity and run-time.

The FEGR approach extracts four features from each vertex within a graph. Through experimentation, we have determined that the following four feature metrics give the best balance between topological sensitivity and run-time. However, other metrics could also be used if other characteristics of a graph are important. For each of the four vertex features listed below, a value is extracted for each vertex $v \in V$:

1) **Node degree** δ: the number of connections a given node has to other nodes in the network.
2) **Clustering coefficient** c: the clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together.

$$c_u = \frac{|(v_1, v_2) \in E : v_1, v_2 \in N|}{(2^d u)}.$$

The numerator in the above equation counts the number of edges between neighbours of node u (where we use $N(u) = \{v \in V : (u, v) \in E\}$ to denote the node

Dataset	SP	NHK	EHK	GK	NetSimile	FGSD	NetLSD		DGSD	**Ours**
							$w(g)$	$h(g)$		
Mutag	86.60	85.06	85.37	77.01	83.42	**88.26**	82.40	83.31	87.70	87.80
PTC	59.00	60.58	57.54	57.56	55.80	60.70	57.22	53.49	**61.32**	58.20
Proteins	74.12	74.29	59.56	73.22	69.71	70.25	68.10	72.14	73.68	**76.10**
NCI1	71.65	75.52	50.04	58.12	68.87	**79.75**	61.94	67.25	73.48	78.9
NCI109	71.48	75.23	50.37	58.97	67.45	**80.44**	60.38	64.64	72.01	77.67
AIDS	99.24	99.2	99.60	98.75	97.95	98.5	93.7	99.69	**99.8**	**99.8**
D&D	77.94	75.81	58.65	>D	73.86	75.9	70.21	72.33	78.52	**78.54**

TABLE I: Classification accuracy comparison. >D indicates experimental time exceeds a day. We report the state-of-the-art results from DGSD because we set our experimental procedure similar to it to provide a fair comparison.

neighbourhood). The denominator calculates how many pairs of nodes there are in the of neighbourhood u.

3) **the average degree of neighbours** k: it is the average degree of the neighbourhood of each node.

$$k_{nn,i} = \frac{1}{|N(i)|} \sum_{j \in N(i)} k_j$$

where $N(i)$ are the neighbours of node i and k_j is node the degree of node j which belongs to $N(i)$.

4) **Average clustering of neighbourhood**: The average clustering score of the neighbourhood is taken for each vertex by taking the mean of all the local clustering scores for the neighbourhood of the vertex.

2) Global features: To make FEGR sensitive to higher level features, we extract a selection of four global features from the graph. The global features, chosen to represent each graph, were selected due to their ability to capture key elements of global graph topology, whilst being efficient to compute. We consider the following global features:

1) The largest five eigenvalues $\{e_1, \ldots, e_5\}$. We keep the top five values as these have been previously shown to be the most significant for predicting the properties of a system (e.g., in control theory, the largest eigenvalue can be used to predict stability, while the second value has a variety of applications [50]). Keeping the largest five keeps the fingerprint the same size for all graphs.
2) Total number of nodes, $N = |V|$.
3) Total number of edges, $L = |E|$.
4) Number of connected components, C. This is the total number of components within the graph, with a component being a subgraph in which there is a possible path between every vertex, whilst vertices in different components have no possible path between them.

To encode the local and global properties of the graph, we initially compute the local features for each vertex. Statistics are then computed for each of the four local features. We use the statistics of median, mean, standard deviation, skewness, and kurtosis to form the graph local descriptor. These can be concatenated into a vector of twenty values:

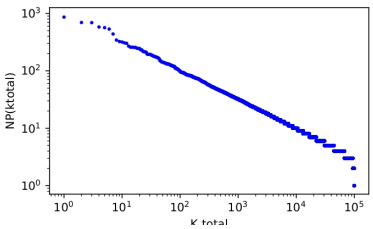
$$\{med(\delta), \bar{\delta}, sd(delta), skew(\delta), kur(\delta), \ldots, med(n), \bar{n}, sd(n), skew(n), kur(n)\}$$

where $med(\square), \bar{\square}, sd(\square), skew(\square)$ and $kur(\square)$ are the median, mean, standard deviation skewness and kurtosis of $\square$. We can then concatenate the four global features to this fingerprint:

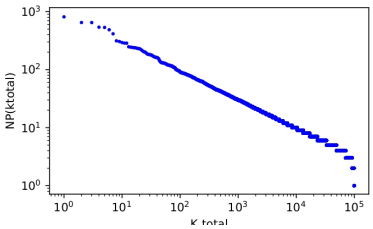
$$\{med(\delta), \bar{\delta}, sd(delta), skew(\delta), kur(\delta), \ldots, med(n), \bar{n}, sd(n), skew(n), kur(n), e_1, e_2, e_3, e_4, e_5, N, L, C\}.$$

Local-level statistics help to distinguish between graphs where the global-level features are the same, and vice versa. However, if both local- and global-level statistics are the same between two graphs, we need more information in order to determine if they are truly the same or different. We argue that the properties of a node can be used to make this distinction.

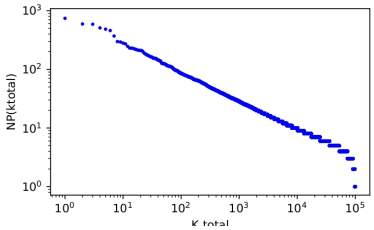
3) Aggregated Node Properties: Node properties represent the data held by a node. For example, in citation networks such as Cora [51], node properties are the word embedding of the abstracts and titles of the corresponding papers. Thus, properties of graph nodes carry important information that could help in the machine-learning task. In general, these node properties can be arbitrary information. This needs to be converted into a fixed-length (numeric) vector which represents the node properties. This can be achieved by many approaches and is bespoke to the problem space at hand. We therefore assume, without loss of generality, that the properties for node i can be mapped to a vector of values $[x_{i,1}, x_{i,2}, ..., x_{i,n}]$. This gives us a vector of length n for each of the m nodes in the graph. To incorporate property information in our graph descriptor, we need to aggregate this data down to a simple vector. We could aggregate this down using the same approach as used for the local node properties (median, mean, standard deviation, skew and kurtosis for each of the n elements); however, initial results where we just sum the vectors have given promising



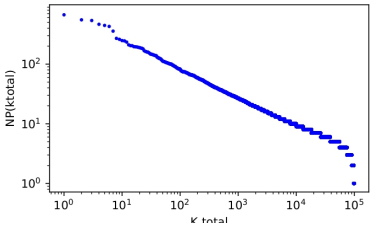
(a) Rewiring 20K



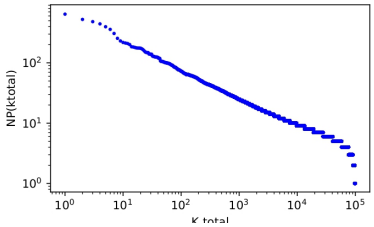
(b) Rewiring 40K



(c) Rewiring 60K



(d) Rewiring 80K



(e) Rewiring 100K

Fig. 2: Sensitivity to Rewiring Edges

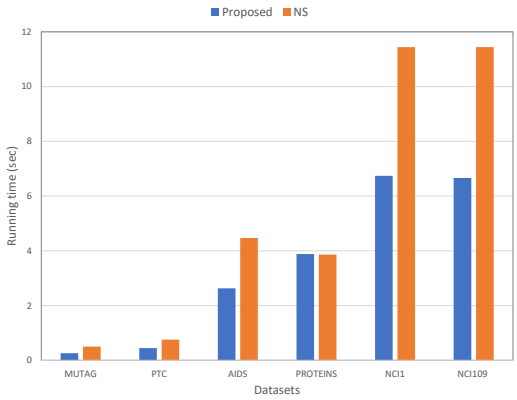


Fig. 3: Run-time comparison of the proposed method against NetSIMILE (NS) on graph classification datasets

results. Therefore, we define the aggregate properties as

$$[\sum_{i=1}^{m} x_{i,1}, \sum_{i=1}^{m} x_{i,2}, ..., \sum_{i=1}^{m} x_{i,n}].$$

This properties fingerprint can now be concatenated with the local and global statistics and used for downstream machine-learning tasks.

V. Experimental Setup and Results

In this section, we provide a comparison of the proposed framework in a graph classification setting with state-of-the-art methods. Mainly, we focus on the experimental setting of DGSD [1] to compare with. DGSD is a recently-proposed graph descriptor that extracts graph representation in both centralized and distributed environments by using very simple graph statistics. The nature of our proposed work is quite similar to this method; therefore, we adopt the exact experimental setting utilised in DGSD and compare our results with it and all other methods reported in DGSD. Similar to DGSD, we use Random Forest algorithm with 10-fold cross-validation and grid search. We kept all the other parameters the same as suggested in DGSD.

A. Datasets

For graph classification, we consider bioinformatics datasets including MUTAG, PTC_MR, PROTEINS, NCI1, NCI109, AIDS, and DD. The number of classes in these datasets is two. These are benchmark graph classification datasets available on Torch Geometric website and presented in [10].

B. Baseline

For baselines, we consider recent graph representation methods, DGSD [1], NetLSD, heat $h(g)$ and wave $h(w)$ kernels [24], FGSD [27], statistical method NetSimile [52] and well-known graph kernels including Shortest Path [26], Neighbourhood Hash Kernel (NHK) [53], Edge Histogram Kernel (EHK) [54], and Graphlet Sampling Kernel (GK) [55].

C. Results

We can observe from the results (Table I) that the proposed method outperforms all other methods on PROTEINS and DD datasets while the results on the AIDS dataset are identical to that of DGSD. We outperform all other methods on AIDS as well. On the remaining dataset, we achieved results within 1% of the top result on MUTAG, and NCI1 and within 3% on PTC and NCI109 datasets. These results clearly show that graph global features positively contribute to the graph representation.

1) Method Sensitivity Evaluation: Sensitivity to change in a network is quite an important property of any graph descriptor. It means that if the topology of a graph is slightly disturbed, then the corresponding graph representation should also change accordingly. To evaluate this, we consider synthetic experiments where we generated a graph with Barabasi Abert model with 100,000 nodes and 300,000 edges. Then we consider the process of randomly rewiring the network. By

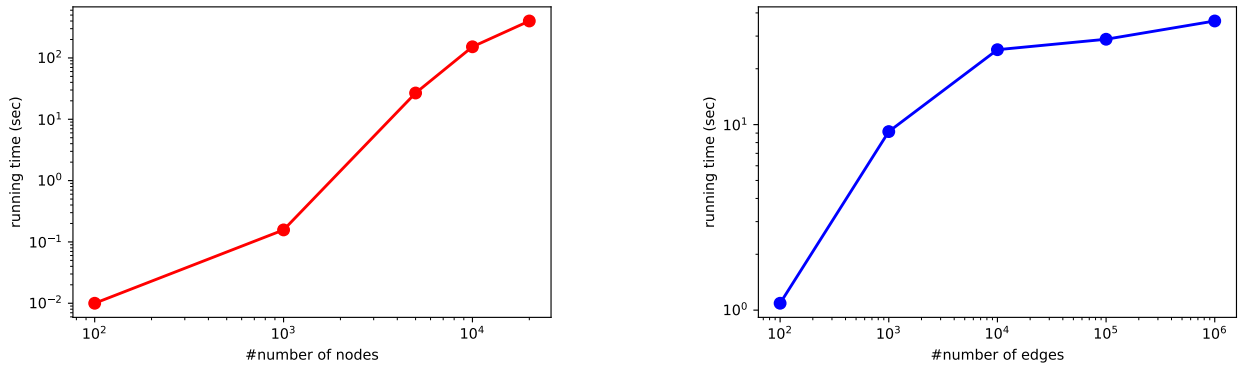


Fig. 4: Run time analysis on the increasing of number of nodes and edges in a graph

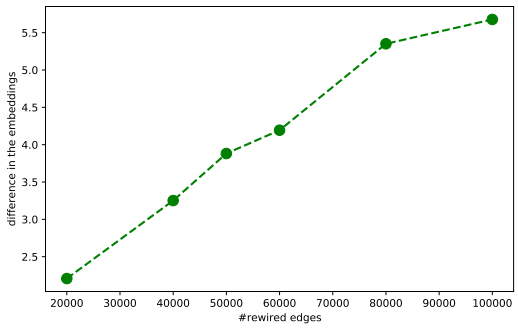


Fig. 5: Sensitivity of our method to the graph topology

random rewiring, we mean that we randomly pick an edge, remove its starting and ending nodes, and connect that edge with two other random nodes.

We consider six network rewiring experiments, which are rewiring $20,000$, $40,000$, $50,000$, $60,000$, $80,000$, and $100,000$ edges in the previously generated original graph. We show the degree distribution of these graphs in Figure 2. To find the distance between the embedding of the original graph and the rewired graph, we use Canberra distance. Also, we show the difference in the embeddings in Figure 5. We can clearly see that as the number of edges in the rewiring increases, the distance between the embeddings increases. This asserts the proposed descriptor is very sensitive to the graph topology, which means it can discriminate between similar graphs very well.

2) Running Time Analysis: We also evaluate the proposed method in terms of running time against NetSimile and consider three different experimental settings. In the first setting, we consider five benchmark graph classification datasets: MUTAG, PTC, PROTEINS, AIDS, NCI1, and NCI109 and run both methods. We show the run-time comparison in Figure 3. We can see that the proposed method takes less time than NetSimile on each dataset. Specifically, it takes half the time on NCI1 and NCI109 datasets. In the second experimental setting, we consider simulation on an increasing number of nodes and explore the run times. For that, we consider Erdős–Rényi random graphs of size $\{100, 1,000, 5,000, 10,000, 20,000\}$. Similarly, in the third experiment, we consider increasing the number of edges in graphs of the same size. For that, we consider a graph of size 5000 (number of vertices), and the number of edges is chosen from the set $\{10^2, 10^3, 10^4, 10^5, 10^6\}$. We apply the proposed method to these graphs and report the run-time as shown in Figure 4. We can see an increase in the run-time when either nodes or edges is increased in the graph. Overall, it processes a graph with a million edges in 36 seconds. These results clearly demonstrate that the proposed method is scalable on sufficiently large graphs.

VI. Conclusion

This paper presents a novel graph representation methods for a graph classification problem. It primarily explores various graph theoretic frameworks for this problem and concludes with nine measures to construct a graph descriptor. These measures capture both local and global level information of the graph. Along with the theoretical features, this paper also leverages aggregated node features to get an expressive representation. Evaluation of a number of graph classification datasets shows the effectiveness of the proposed method.

References

[1] A. Said, S.-U. Hassan, S. Tuarob, R. Nawaz, and M. Shabbir, "Dgsd: Distributed graph representation via graph statistical properties," *Future Generation Computer Systems*, vol. 119, pp. 166–175, 2021.

[2] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[3] A. Said, T. D. Bowman, R. A. Abbasi, N. R. Aljohani, S.-U. Hassan, and R. Nawaz, "Mining network-level properties of twitter altmetrics data," *Scientometrics*, vol. 120, no. 1, pp. 217–235, 2019.

[4] A. Said, S.-U. Hassan, W. Abbas, and M. Shabbir, "Netki: A kirchhoff index based statistical graph embedding in nearly linear time," *Neurocomputing*, 2020.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[6] H. Jia, S. Ding, X. Xu, and R. Nie, "The latest research progress on spectral clustering," *Neural Computing and Applications*, vol. 24, no. 7, pp. 1477–1486, 2014.

[7] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[8] A. Said, M. U. Janjua, S.-U. Hassan, Z. Muzammal, T. Saleem, T. Thaipisutikul, S. Tuarob, and R. Nawaz, "Detailed analysis of ethereum network on transaction behavior, community structure and link prediction," *PeerJ Computer Science*, vol. 7, p. e815, 2021.

[9] S. Bonner, J. Brennan, I. Kureshi, and A. S. McGough, "Efficient comparison of massive graphs through the use of'graph fingerprints'," in *12th International Workshop on Mining and Learning with Graphs*. Newcastle University, 2016.

[10] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[11] D. Jiang, Z. Wu, C.-Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu, and T. Hou, "Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models," *Journal of cheminformatics*, vol. 13, no. 1, pp. 1–23, 2021.

[12] S. Uddin, A. Khan, M. E. Hossain, and M. A. Moni, "Comparing different supervised machine learning algorithms for disease prediction," *BMC medical informatics and decision making*, vol. 19, no. 1, pp. 1–16, 2019.

[13] J. Schrodt, A. Dudchenko, P. Knaup-Gregori, and M. Ganzinger, "Graph-representation of patient data: a systematic literature review," *Journal of medical systems*, vol. 44, no. 4, pp. 1–7, 2020.

[14] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE transactions on systems, man, and cybernetics*, no. 3, pp. 353–362, 1983.

[15] M. R. Garey and D. S. Johnson, "Computers and intractability, vol. 29," 2002.

[16] C.-L. Lin, "Hardness of approximating graph transformation problem," in *International Symposium on Algorithms and Computation*. Springer, 1994, pp. 74–82.

[17] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning theory and kernel machines*. Springer, 2003, pp. 129–143.

[18] R. Kondor and H. Pan, "The multiscale laplacian graph kernel," *Advances in neural information processing systems*, vol. 29, 2016.

[19] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[20] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.

[21] D. Koutra, J. T. Vogelstein, and C. Faloutsos, "Deltacon: A principled massive-graph similarity function," in *Proceedings of the 2013 SIAM international conference on data mining*. SIAM, 2013, pp. 162–170.

[22] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, "Web graph similarity for anomaly detection," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 19–30, 2010.

[23] J. Bento and S. Ioannidis, "A family of tractable graph distances," in *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018, pp. 333–341.

[24] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, "Netlsd: hearing the shape of a graph," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2347–2356.

[25] T.-S. Kuo, K.-S. Tseng, J.-W. Yan, Y.-C. Liu, and Y.-C. Frank Wang, "Deep aggregation net for land cover classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 252–256.

[26] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 2005, pp. 8–pp.

[27] S. Verma and Z.-L. Zhang, "Hunt for the unique, stable, sparse and fast feature learning on graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[28] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[29] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "Shape google: Geometric words and expressions for invariant shape retrieval," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 1, pp. 1–20, 2011.

[30] R. Gal, A. Shamir, and D. Cohen-Or, "Pose-oblivious shape signature," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 2, pp. 261–271, 2007.

[31] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 807–832, 2002.

[32] M. M. Bronstein and A. M. Bronstein, "Shape recognition with spectral distances," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 1065–1071, 2010.

[33] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer graphics forum*, vol. 28, no. 5. Wiley Online Library, 2009, pp. 1383–1392.

[34] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE, 2011, pp. 1626–1633.

[35] K. Kloster and D. F. Gleich, "Heat kernel based community detection," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1386–1395.

[36] W. L. Hamilton, Z. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, pp. 52–74, 2017.

[37] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.

[38] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, no. 2005, 2005, pp. 729–734.

[39] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[40] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[41] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *International conference on machine learning*. PMLR, 2018, pp. 1106–1114.

[42] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[43] Y. Chen, G. Ma, C. Yuan, B. Li, H. Zhang, F. Wang, and W. Hu, "Graph convolutional network with structure pooling and joint-wise channel attention for action recognition," *Pattern Recognition*, vol. 103, p. 107321, 2020.

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

[45] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[47] S. Bonner, A. Atapour-Abarghouei, P. T. Jackson, J. Brennan, I. Kureshi, G. Theodoropoulos, A. S. McGough, and B. Obara, "Temporal neighbourhood aggregation: Predicting future links in temporal graphs via recurrent variational graph convolutions," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5336–5345.

[48] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[49] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[50] S. K. Simić, M. Anelić, C. M. da Fonseca, and D. Živković, "Notes on the second largest eigenvalue of a graph," *Linear Algebra and its Applications*, vol. 465, pp. 262–274, 2015.

[51] B. London and L. Getoor, "Collective classification of network data." *Data Classification: Algorithms and Applications*, vol. 399, 2014.

[52] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos, "Network similarity via multiple social theories," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2013, pp. 1439–1440.

[53] S. Hido and H. Kashima, "A linear-time graph kernel," in *Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 179–188.

[54] M. Sugiyama and K. Borgwardt, "Halting in random walk kernels," *Advances in neural information processing systems*, vol. 28, 2015.

[55] N. Pržulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, 2007.