Interpretable Multi-Scale Graph Descriptors via Structural Compression

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

Abstract—Compact graph representations that preserve relevant topological information allow the use of rich machine learning toolset for a data-driven graph analytics. Some notable graph representations in the literature, especially ones based on spectral features, are fairly fruitful in their respective applications but they either lack interpretability or are unable to effectively encode a graph's structure at either local or global scale. In lieu of this, we propose the Higher-Order Structure Descriptor (HOSD): an interpretable graph descriptor that captures information about the patterns found in a graph at multiple scales. Scaling is achieved using a novel graph compression technique that reveals successive higher order structures. Proposed descriptor is invariant to node permutations due to its graph-theoretic nature. Moreover, any representation produced by HOSD is readable in language of graph properties, and is, thus, helpful towards explainable decision making. We analyze HOSD algorithm for time complexity and also prove NP-Completeness of three interesting graph compression problems. A faster version, HOSD-Lite, is also presented to approximate HOSD on denser graphs. HOSD and HOSD-Lite are evaluated on multiple real-world datasets for applicability to classification problems. Results demonstrate that a simple random forest setup based on our representations competes well with the current state-of-the-art graph classification methods.

Index Terms—Graphs, Hierarchical representations, Classification

1 Introduction

To be made Changes
Changes done

- introduction
- related work
- interpretability
- table, GCN results
- authors, affiliations, acknowledgments

On account of their discrete nature and efficacy in representing relationships between pairs of entities, graphs are used to model a plethora of practical structures such as chemical compounds, communication networks, multirobot systems, and co-authorship networks. For more applications, we direct the reader to a comprehensive survey on the representation of data in the form of graphs by Angles et al. [1]. Graphs combinatorial properties and algorithmic aspects have be active topics of research for decades in the computer science community. In recent years, researchers have aggressively looked into data-driven approaches to solve practical problems on graphs. In particular community detection, graph classification, link prediction, node classification, and recommendation systems in networked settings have been studied in context of graphs. Among these problems, graph classification has, arguably, received most attention [2], [3]. The notion of graph classification entails determining which given category a graph belongs to. Consider two or more sets of graphs, where each set corresponds to a category. A graph classification paradigm essentially learns the distinguishing features between these sets. Once trained, it should then be able to accurately determine which category a new, unseen graph belongs to. There are four main challenges in transferring existing machine learning technology to graphs:

- Graphs in a dataset may have arbitrarily different sizes, both in terms of vertices and edges. This is a serious complication because traditional classifiers expect objects of fixed sizes as input. Even nontrivial ways of stretching and compressing graphs to a uniform size have resulted in significant loss in topological information [4].
- In many examples, essential information regarding a network lies with presence or the absence of edges and vertices themselves do not carry any meta-data. In such cases of unlabeled graphs, vertices with similar local neighborhood structure are practically indistinguishable from each other. As a result, one observes many seemingly different looking networks that are, in essence, copies of the same graph with different orders on a vertex set. This is the classical graph isomorphism problem:

  Problem 1 (Graph Isomorphism Problem)**.** *Given two graphs $G = (V, E)$ and $G' = (V', E')$ on n nodes each, does there exist a mapping $\phi : V \to V'$ such that $(u, v) \in E$ if and only if $(\phi(u), \phi(v)) \in E'$?*

  Though not known to be `NP-Hard`, the notorious Graph Isomorphism(GI) problem has evaded a polynomial algorithm so far [5].
- A complete graph classification paradigm should be able to encapsulate information at multiple scales. To elaborate, Kondor et al. give the example of a molecular graph, stating that local features encode information about the presence of different functional groups in a molecule, which global features describe how these groups interconnect within the molecule [6].

- *M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

Manuscript received April 19, 2005; revised August 26, 2015.

- Graph classification solutions also suffer from the lack of *interpretability*. State-of-the-art methods for graph classification are based on spectral or data-driven representations that offer very little to none in terms of explainability. For an end-user of graph analytics, these "magical numbers" shroud classification results in an extra layer of undesired mystery.

In this work, we aim to design a graph representation to address the above-mentioned challenges. Our approach is to effectively encode the structural information prevalent within a graph. In this vein, the embeddings we construct are based purely on topological information within the given graph, and corresponding metadata, including attributes or labels, is ignored. We propose, the Higher-Order Structure Descriptor (HOSD), a vector representation of graphs to solve the graph classification problem. HOSD captures neighborhood structure of graph at local and global scale and is, therefore, suited for a variety of applications.

HOSD is able to capture topological information at local and global scale by considering "zoomed out" or "compressed" forms of the input graph. We define *Familial Graph Compression (FGC) problems* to take "snapshots" of the graph at multiple levels. Copies of a given family of subgraphs are repeatedly "compressed" to construct a smaller graph that represents a higher order structure. This is illustrated in Figure 1. We prove `NP-Hardness` of three different instances of familial compression problems which shows that there are no polynomial time solutions for such problems in general unless `P=NP`. On the other hand, we observe that FGC is, in practice, efficiently solvable for a small yet practically important problem instances.

The dimension of HOSD representation is fixed for a given dataset, i.e., graphs of different sizes are all mapped to same space so that standard classification methods can be employed. HOSD is permutation invariant in the sense that a graph and its variants with permuted vertices but same topological structure will produce the same representation. Another important advantage of HOSD is its readily available interpretability. While interpretability may not have any effect on the classification accuracy, it helps the end-user understand the choices made by the machine.

2 Related Work

Graph classification has high impact applications in bioinformatics, social network analysis, formal methods etc., and thus has attracted researchers from all these areas. Machine learning related relevant literature on this problem can, loosely, be divided into three categories: 1) *Graph Convolutional Neural Networks*, 2) *Graph Kernel Methods*, and 3) *Graph Embeddings*. Other than these three sections, in *Graphlet Counting* section we have discussed methodologies related to graphlet counting but these techniques not directly applied to graph classification. Related work affiliated with all these three categories is discussed below.

2.1 Graph Kernels

In this section we discuss different graph kernels which are primarily used for graph classification. Graph kernels are R-convolution kernels [7] on pairs of graphs that essentially compute similarity measures between every pair of graphs in a given set based on the graph theoretic properties. Computing complete graph kernels is thought to a hard problem as this reduces to deciding whether two graphs are isomorphic. However, notable graph kernels with reliable accuracies have been proposed.

First notable work in this direction was by [8]. In unlabelled graphs, kernel is computed by comparing shortest-paths between every pair of nodes based on their lengths. Kernel is equivalent to computing product of Weiner Indices of the two graphs [9]. Computing shortest-path kernel value for a pair of graphs takes $O(n^4)$ where n is total number of nodes in two graphs. Similar kernels are known that are based instead on subtree structures [10] edit-distance [11].

The Graphlet Kernel presented in [12], compare the distribution of induced subgraphs (called *graphlets*) of fixed size k in two graphs. Since runtime is exponentially dependent upon value of k, this can only be implement for small graphlets. It is similar to our methodology in counting subgraph/graphlet of size $k \in \{3, 4, 5\}$. These solutions will not likely be able to encode properties of sub-structures in graphs on larger scale [13].

Random walk kernel [14] a kernel used for classification based on random walk has been introduced. Fast subtree kernel based on WL test of isomorphism has been studied before the introduction of WL kernels [15] and then the same author introduced WL kernel in 2011. Weisfeiler-lehman graph kernels [16] as they have reported state of the art accuracies for NCI1 and NCI109. The Weisfeiler-Lehman Kernel [16] is based on Weisfeiler-Lehman graph isomorphism test [17] has been studied. The WL kernels might have better accuracy because they utilize node labels and even for the unlabelled nodes they use node degree as node labels. WL kernels are not scalable, as reported that for the dataset DD it took 3 days for the WL edge kernel and, it took more than a year for the WL shortest path kernel to complete the execution. WL kernel can only utilize discrete node labels. [18] presents kernel which is scalable and can use vector or continuous node attribute for graph classification. Variants of WL kernel have also been introduced which tries to balance between capturing the information between local and global structural properties of graph [19]. They also argue that multi-dimensional WL kernel ($hparameter$) can exponentially increase the complexity of the problem. In [20] the authors introduces propagation kernels that uses node labels that can handle vector and continuous node labels. Another tree based subgraph kernels has been introduced whose methods gauge depth information through a family of K-layer expansion subgraphs rooted at a vertex [21]. Kernels defined on unlabeled graphs which capture global properties of graphs using the Lovasz number and its associated orthonormal representation are studied as well [22] but their reported accuracy on benchmark datasets are not up to the latest state of the art results excpet for the dataset MUTAG.

Kernel based on the Jensen-Tsallis (JT) q-differences between probability distributions over the graphs has been introduced and their results show high accuracy on NCI1 and NCI109 [23]. Deep graph kernel [24] have also compared WL kernel with their work using $h = 2$. Deep graph kernel have reported much higher accuracy for $h = 2$ as compared to the FGSD. Deep graph kernel use neural networks to learn latent representation of subgraphs. In case of WL kernel deep graph kernel uses multiset of nodes to learn the representation. Propogation kernel [25] extension of their previous work [20] uses distributions from random walk to capture structural information encoded in node labels, attributes, and edge information they are based on older methods but they claim that their method is fast. Multiscale laplacian kernel [6] which has been recently introduced in the field of graph kernels

shows that it can encode graph structural information on multiple scales. In [26] a pyramid kernel has been introduced to capture global scale information of the graph. Their own method may not be able to give state of the art result but their kernel combined with WL kernel produced some quite impressive graph classification results. Two new kernels have been introduced in the field of chemoinformatics The first kernel is based on the notion of edit distance while the second is based on subtrees enumeration [27]. It seems that substructure counting is quite popular and well studied field in chemoinformatics or bioinformatics . [28] studied an enhancement technique for kernels that takes point projected by kernel to a new space where the distance between the different classes if much greater, however it is an enhancement technique which is build upon kernels and it adds nothing new to embedding or encoding features graph.

Kernels do not scale well as they require expensive computation. Moreover, most kernels do not incorporate information at different scales. One exception to this is the Multiscale Laplacian Kernel [6] which builds a hierarchy of nested sub-graphs. They propose a multi-scale kernel between two input graphs. But their kernel is spectral in nature and builds on eigenvalues of subgraphs "centered" at a vertex of varying sizes.

2.2 Graph Convolutional Networks

Inspired by the success of convolutional neural networks (CNNs) in computer vision, spatial-based GNNs learn to make node level representations by convolving the information from said node's neighborhood. Some GNNs also emulate CNN pooling layers by down-sampling graphs to coarser sub-structures. These node representations can be then aggregated to construct a representation for the entire graph.

Graph CNNs learn to classify data via convolutions over the graph or their Laplacian. WL embedding to feed it into the graph convolutional network for graph convolutional network [4]. Similar work have been done to get the embeddings using kernel and then feeding them into convolutional neural network for graph classification [29]. Diffusion convolution neural network has been introduced for graph classification [30]. They propose a method 'diffusion-CNN' that can be applied to small graphs. Their idea is based on 'diffusing' node features and adjacency matrix to build a weighted network at each layer of neural architecture; interestingly, their architecture have no pooling operation. Variation of convolutional neural network has been studied which include ranking neighbors before aggregating information [31] incorporating different weights based on the shortest-path distance from a node [32], and introducing extra parameters at each convolution layer to better aggregate the previous layer's output alongside concatenating the aggregated outputs from each layer to construct the final graph representation [33]. They aggregated embeddings corresponding to subgraph and neigbhourhood information for classification. Recurrent neural network is also used to focus on on specific structure of graphs for classification [34]. Morris et al. showed that graph neural networks embody the same prowess as WL-Kernels, with the added benefit of adapting to the classification task at hand [35]. They also present k-GNNs, a k-WL inspired generalization of the standard GNN model, allowing one to capture information at multiple scales. In [36], [37], while using convolutional networks, each graph is treated as a signal on which convolutional filters and pooling has been applied for feature extraction. "Semi-supervised classification with graph convolutional networks" [38] have not extended their worked on graph classification rather they have discussed their work for node classification.

Convolutional networks also have been used to extract interpretable molecular feature from chemical dataset represented as graphs [39] but they have not been tested on graph classification. We refer the reader to Zonghan et al. for a survey on the topic [40] we also refer to [41] for detailed study of graph embedding which also include deep-learning based method for graph classification. Deep learning based approaches relies on improving the pipeline of neural network with innovative operators and filters that work well with graphs. Most of them use edge and node labels as they are data driven and deep-learning based models need more data for better performance GCN approaches are data-driven, and usually require labels on the nodes or edges to achieve high accuracy. Like all CNN-based methods, interpretability of GCN is still an open problem.

2.3 Graph Embeddings

A graph embedding refers to a vector representation of a graph in some fixed dimension d. One could bridge the gap between classical machine learning classification paradigms and graph classification by constructing viable vector descriptors. However, converting graphs into vectors is a difficult task as elaborated in the last section. One way to cater to this issue would be to make a descriptor using graph-theoretic properties such as the number of nodes, average degree [42], or the presence of specific sub-structures. Graphlet spectrum [43] has been presented which is a system of graph invariants derived by means of group representation theory that capture information about the number as well as the position of labeled subgraphs in a given graph and their results are quite poor as compared to the current state of the are results because the paper was published in 2009. Graph classification via topological and label attributes has been studied and their results are not that great except for MUTAG and PTC_MR [44]. Dutta et al. [45] propose using random walks coupled with hashing to capture the presence of different subgraphs in a graph. However, these approaches are unable to capture structural information about the graph at higher scales. Tsitsulin et al. [46] and Verma et al. [47] propose the use of spectral features, which they believe are able to encapsulate properties of a graph at different scales. While all of these approaches successfully make viable Euclidean embeddings for graphs, these representations lack the interpretability that would help an individual using these methods discern exactly what kind of sub-structures distinguish between categories of graphs.

2.4 Graphlet Counting

Counting graphlet of size 4 on large graphs have been studied in [48] and graphlet counting for size $4, 5$ presented as combinatorial problem has been studied in [49]. Both of the previous methods have not extended their work on graph classification. Comparing networks using graphlet distribution has been done before [50]. It is based on graphlet counting. They have developed a framework which measures the number of nodes touching k edges, into distributions measuring the number of nodes touching k graphlets and according to them graphlets are small connected non-isomorphic subgraphs of a large network. They have not tested their method on graph classification. Similar study have been done to mine discriminative frequent subgraphs

for graph classification [51]. Also another similar technique for efficient graphlet (of size $3, 4$) counting for large graphs has been done [52]. Please see [53] for detailed discussion about graphlet counting in the field of bioinformatics which formulates the chemicals as graphs. Please see recent surveys on this topic here [3] and [2] for graph embedding and their applications.

3 Higher-Order Structure Descriptor

In this section, we outline how to construct a permutation invariant representation of a graph that is well-suited as an input to any classification algorithm. Before we could describe proposed descriptor, we need to define Familial Graph Compression problem. This graph compression method will be useful to construct abstract graphs that represent higher order structure of the original graph. We start by laying out some preliminary notations and definitions.

3.1 Preliminaries

Input to a graph classification problem is a (potentially infinite) multiset of graphs where each graph G is a collection of vertices V and edges $E \subseteq V \times V$ i.e. pairwise interactions between pairs of vertices. These graphs are allowed to be of variable orders (the number of nodes), and sizes (number of edges). For a vertex u, neighborhood $N(u)$ is defined as the set of all vertices $v \in V$ such that (u, v) is an edge in E. Degree $d(u)$ is defined as the size of neighborhood of a vertex u. We assume, for the sake of simplicity, that each G is undirected and unweighted, i.e. for $u, v \in V$, an edge (u, v) is same as the edge (v, u), and there are no weights or any other meta-data associated with any vertex or edge. For a fixed graph $G = (V, E)$, a given $F = (V_F, E_F)$ is called a *motif* of G, if F is isomorphic to a subgraph in G i.e. F is a motif if there exist $V' \subset V$ and a function $\phi : V_F \to V'$ such that for all edges $(u, v) \in E_F$ there is an edge $(\phi(u), \phi(v)) \in E$. Similarly, $F = (V_F, E_F)$ is called a *graphlet* of G, if F is isomorphic to an *induced* subgraph in G i.e. F is a graphlet if there exist $V' \subset V$ and a function $\phi : V_F \to V'$ such that for all edges $(u, v) \in E_F$ *if and only if* there is an edge $(\phi(u), \phi(v)) \in E$. We will use the term motif (and similarly graphlet) for both F and any of its isomorphic copies in G.

For a given equivalence relation $\sim$ on the set vertices of a graph G, the quotient graph, denoted by $G/\sim$, is a graph where vertex set is the set of equivalence classes defined by $\sim$ and there is an edge between a pair of vertices (classes) if and only if there is an edge between any pair of vertices of two corresponding classes in G. Intuitively, in quotient graphs prescribed subsets of vertices are merged together and incidence is preserved without creating multi-edges [54]. We will repeatedly deal with graphs with names G, H, and F_i; if not explicitly mentioned otherwise, their vertex and edge set will, respectively, be denoted by (V_G, E_G), (V_H, E_H) and (V_{F_i}, E_{F_i}). Finally, for a set V and a positive integer c, $\binom{V}{c}$ is defined as the set of all c-size subsets of V where $\left|\binom{V}{c}\right| = \binom{|V|}{c}$. In many applications, there might be labels associated with nodes in the graphs but we choose to ignore such information as we are interested in deducing inferences based only on topological information.

3.2 Familial Graph Compression Problem (FGC)

There is a rich history of problems in graph theory that involve, informally speaking, packing a lot of copies of a small graph in a target graph or, viewed otherwise, decomposing a graph into many smaller subgraphs of a special kind. In familial graph compression, we are interested in the structure revealed after a certain "optimal" packing is carried out. We start by defining an equivalence relation on vertex set V of G based on a motif (or a graphlet) F. Consider the relation R_F where vertex u is related to v whenever u and v lie in a subgraph of G isomorphic to F. We define $\sim_F$ to be the transitive and reflexive closure of R_F. Intuitively, if two motifs share a common vertex in G, then all vertices in both motifs are related in $\sim_F$. Clearly, $\sim_F$ is an equivalence relation on V. Then, an F-**compression step** (or just compression step when F clear from context) is defined as computing the quotient graph $G/\sim_F$. The **Familial compression** of a graph G for a family $\mathcal{F}$ is the process of repeatedly applying F_i-compression step on G where after each step G is replaced by the quotient graph of the previous step. Thus, we say that a graph H can be constructed by $\mathcal{F}$-compression of G if there exist a sequence of graphs: $[G^0 \quad G^1 \quad G^2 \quad \dots \quad G^k = H]$ where $G^0 = G$ and $G^j = G^{j-1}/\sim_{F_i}$ i.e. G^j is result of an F_i-compression on the graph G^{j-1} for some $F_i \in \mathcal{F}$. Note, that a graph H may be constructed in several different ways of varying number of compression steps. To avoid trivial compressions, we restrict that each $F \in \mathcal{F}$ contains at least two vertices. Under this condition, we have the following observation:

Observation 2. *If a graph H can be constructed by $\mathcal{F}$-compression of G then H can be constructed in at most $|V_G| - |V_H|$ compression steps.*

This holds because in each compression step, one must compress number of vertices reduce by at least. The following is the FGC problem:

Problem 3 (Familial Graph Compression Problem)**.** *Given simple graphs G, H and a family $\mathcal{F}$ of motifs(equivalently graphlets), can H be constructed by an $\mathcal{F}$-compression of G?*

It turns out that FGC is intractable in general. In fact, we can show that FGC is `NP-Complete` even for some fixed choices of input graphs.

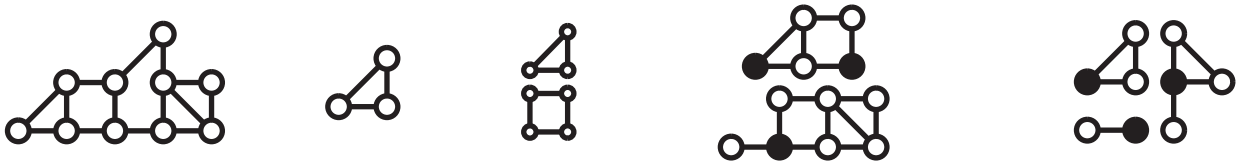
Theorem 4. *The FGC problem is `NP-Complete` when:*

1) *G is simple graph on n vertices, H is the single vertex graph, and family $\mathcal{F}$ contains a single motif C_n i.e. a cycle on n vertices.*
2) *G is a simple graph on $n = 3k$ vertices, H is the single vertex graph, and $\mathcal{F}$ contains a single motif with k disjoint triangles.*
3) *G is a simple graph, H is the single vertex graph, and $\mathcal{F}$ is a family of graphlets.*

The first two statements can be proved using (rather easy) reductions from the `Hamiltonian Circuit` and `Triangle Packing` problems respectively. The third statement is showed by a reduction from the `EXACTLY-ONE-IN-THREE 3SAT` problem. All proofs are discussed in the section 5.

While solving the FGC problem on general is not tractable, we outline here a simple algorithm that, in practice, efficiently solve FGC when $\mathcal{F}$ a small number of constant size motifs and $|V_H| = \Omega(|V_G|)$. Unfortunately, we can't hope to theoretically guarantee a polynomial time solution even for this very restrictive case.

Proposition 5. *FGC problem is `GI-Complete` even when $|V_H| = \Omega(|V_G|)$ and $\mathcal{F}$ contains a constant number of small*



(a) Input graph, G. (b) Target graph, H. (c) Family, $\mathcal{F}$. (d) Level two representations. (e) Level three representations.

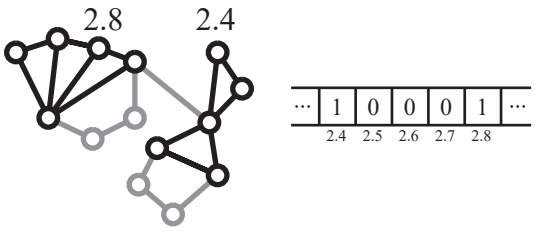
Fig. 1: An instance of the Familial Graph Compression problem. The level two representations have been obtained by compressing instances of member of $\mathcal{F}$ in G. Similarly, level three representations are obtained by compressing instances found in level two representations. Nodes formed via compression are represented as black.

fixed size motifs.

To see this, imagine G and H to be two bipartite graphs of equal size and equal order, and let $\mathcal{F}$ contain a single triangle graph. FGC problem on this instance is equivalent to the *Bipartite Isomorphism problem* which is `GI-Complete` and doesn't yield a polynomial time algorithm unless `GI`$\subseteq$`P` [55]. In our experiments, however, the following algorithm does run efficiently for the above-mentioned case.

3.2.0.1 Algorithm: For the sake of readability, we assume that for all $F_i \in \mathcal{F}$, F_i is a graphlet on c vertices. Discussion below holds even F_i's are of different (but small) orders and/or are treated as motifs instead of graphlets. Observe that a single compression step can be performed in $O(n^c)$ time by first marking all vertices and edges in, at most, $\binom{n}{c}$ c-subsets and then compressing all marked edges in a Depth First Search(DFS) pass. Compressing an edge is equivalent to replacing two endpoint with vertex while preserving incidence. For a family $\mathcal{F}$ that contains k graphs, we thus generate k quotients G/F_i one for each F_i. In most cases, graph G/F_i is fraction of the order of the original graph. We recursively perform $\mathcal{F}$-compressions on all G/F_i. The process is halted when potential quotient graphs are smaller than H with respect to number of vertices. All resultant graphs are matched against H. Figure 1 illustrates this process on a simple graph.

While constructing representation of a given graph, we will perform familial compression on G for a carefully chosen $\mathcal{F}$ but we are going to avoid having to perform the expensive last step of checking graph isomorphism. Instead, we will compute informative statistics on the distribution of graphlets/motifs in compressed graphs as a signature of G.



(a) Triangles in G. (b) Degree histogram.

Fig. 2: This figure illustrates how a histogram of average degrees is constructed. The triangles in (a) have been highlighted in black.

3.3 A Mullti-scale Representation

In this section, we outline the construction of graph representation. We will construct HOSD in multiple rounds that we call *levels* where successively increasing levels present a higher order view. We assume that we are given the input graph G and a suitable family of graphlets $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$. At the first level, HOSD, denoted by $\mathcal{H}$, will consists of the following sequence:$[\mathcal{H}_{1,1} \;\; \mathcal{H}_{1,2} \ldots \mathcal{H}_{1,k-1} \;\; \mathcal{H}_{1,k}]$ where $H_{1,i}$ is the motif (or graphlet) F_i count in G. In level two, we will construct k quotient graphs G/F_i, and count graphlets in all of them and thus report the second level of $\mathcal{H}$, $[\mathcal{H}_{2,1} \;\; \mathcal{H}_{2,2} \ldots \mathcal{H}_{2,k^2-1} \;\; \mathcal{H}_{2,k^2}]$. Since there are k different quotient graphs, and we compute k statistics for each one, second level descriptor will have size k^2. We continue in this fashion for all ℓ levels and in the end, have a descriptor of the form:

$$\mathcal{H} = [H_{1,1} \;\; H_{1,2} \cdots H_{1,k} \;\; H_{2,1} \;\; H_{2,2} \cdots H_{\ell,k^\ell}]$$

In general, a total of $\sum_{j=1}^{\ell-1} k^j$ quotient graphs are obtained, and histograms are constructed for each graph. These histograms can simply be concatenated to construct a descriptor for G. It is easy to see that the choice of graphs in family $\mathcal{F}$, size of this family, and the number of levels ℓ has a significant bearing on accuracy and runtime of this algorithm. In the following, we list down and provide justification for the design choices we make for the efficient computation of the HOSD representation without markedly hampering the classification accuracy.

3.3.0.1 Graphlets/Motif F_i in $\mathcal{F}$: Recall, that for each graphlet in $\mathcal{F}$, we have to perform a search and compress operation in G. This already puts a limit on the choices of such an F_i due to the theoretical lower bound on runtime of search. Obviously, we have to choose smaller graphlets but some trivial choices have to be ignored as well. For example choosing a single edge will always compress a connected graph G to a solitary vertex and is, thus, not useful for such representation. Apart for this, we consider all nontrivial connected graphlets of small sizes that have a distinctive presence in the training sample. A graphlet that is absent from all of the available graphs is a dimensional burden without any contribution to representation sparsity and is, therefore, ignored. A greater number of graphlets in $\mathcal{F}$ is likely to help with the classification accuracy. We choose this number to be as large as possible constrained only by the maximum allowed runtime.

3.3.0.2 Labels of Compressed Vertices: While constructing the quotient graph G/F_i, there is a question of whether model should distinguish between a F_i-compressed vertex and a vertex that was already present in G. There are different ways one can capture this information, which we refer to as the **verbosity** of our descriptor. In this work we explore three different levels of verbosity for a representation: all vertices are indistinguishable (*Verbosity 1*), there are two types of vertices, compressed and not-compressed, and motifs (or graphlets) containing only not-compressed vertices are counted separately from those which

contain at least one compressed vertex (*Verbosity 2*), graphlets containing varying number of compressed and not-compressed vertices are counted separately as well (*Verbosity 3*). To keep the descriptor relatively simple, and to avoid overfitting our information, further levels of verbosity are not explored. Each level of verbosity has been explored when evaluating HOSD in Section 6.

3.3.0.3 Degree Histogram: To allow HOSD to differentiate between different graphs that appear the same after compression, a histogram of the average degrees of each disjoint compressed structure is concatenated to the final descriptor. For simplicity, the bin-width of this histogram is set to 0.1. This is illustrated in Figure 6.

3.4 Time Complexity

Compressing a graph on n nodes requires searching all (induced) subgraphs isomorphic to pattern graph F_i. For a constant size F_i, this will take $O(n^c)$ for some constant in the worst case. For k subgraphs in family $\mathcal{F}$, and ℓ levels of compression, we construct at most $\sum_{j=1}^{\ell-1} k^j$ compressed graphs. Therefore, constructing these graphs takes $O(k^\ell \times n^c)$. Counting occurances of motifs and/or graphlets can also be performed same amount of time. The degree histograms can be computed in linear time in the size of graph. Overall, runtime is bounded by $O(|V_G|+|E_G|+k^\ell \times n^c)$, for a small constant c.

3.5 Interpreting HOSD

As per the construction of our descriptor, each bin corresponds to presence of a reproducible structure/pattern prevalent in the original graph. Analysis of these bins enables us to learn about different topological structures present in multiple classes. For instance, consider again the example in Figure **??**. HOSD was able to identify the disjoint nature of rings on five and six nodes because when six-rings were compressed, five-rings were only found in the graphs of class B. We believe that such information will aid in finding out properties of chemical compounds or networks, and thereby allow the expansion of knowledge in these domains.

Using HOSG we are able to encapsulate two kinds of information in our histogram as we propagate through the layers: the presence of sub-graphs within other sub-graphs, and the spatial correspondence between disjoint sub-graphs. Notice that as we increase the number of layers, we are able to capture the spatial correspondence of more sub-graphs due to compression at higher order layer. Graphs may have many prevalent sub-structures that may be compressed to reveal important information. At each level graph gets a new representation that is a higher level representation encoding lower (sub-structural) information in it. We can go to arbitrary higher order level to obtain the desired graph representation that can is interpretable because of the easy-to-understand properties/vector the hosd provides for each level. As HOSD propagates through each layer, we are able to encode the presence of larger sub-structures. These larger sub-structures are essentially conjoined combinations of the structures present in $\mathcal{F}$, as illustrated in Figure 3 using a 4-star and triangle. In Figure 3 once triangles are compressed it is easier to distinguish between the the different graphs. For large graphs distinguishable representation can be obtained by compressing it through multiple layers. The vector obtained will encode the histogram of sub-structures at different scale and one can plot the histogram of particular instance lets say triangle to see its distribution on different scale or order.

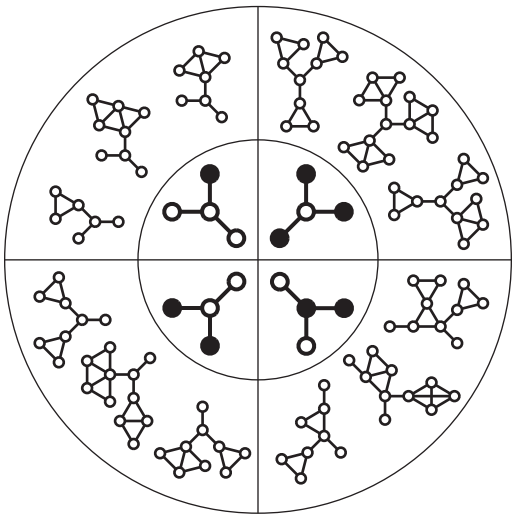


Fig. 3: Possible sub-structures that we can identify by finding instances of 4-stars nodes after compressing triangles. The inner circle includes graphs we may find after compressing triangles, the bold nodes indicate the where the triangle has been compressed. The outer arcs include the possible original structures that we are able to find implicitly.

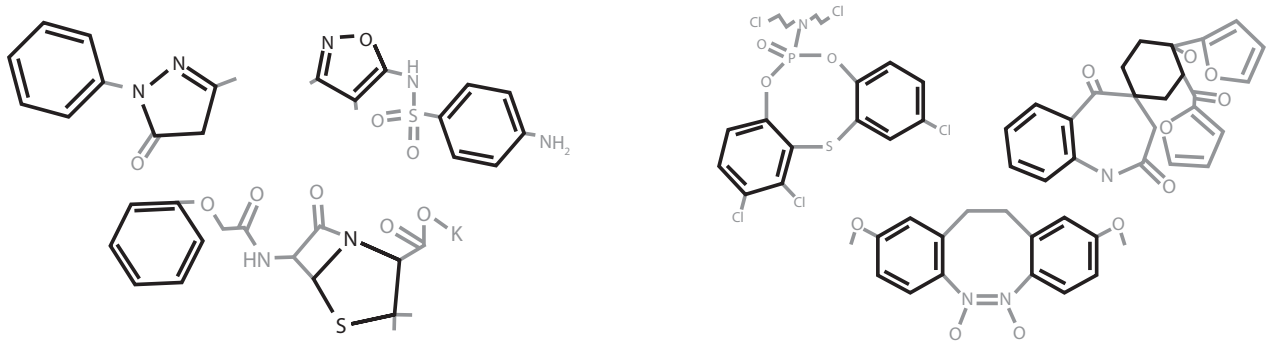
Let's say we want to classify communities whose interactions typically look like cycles. As illustrated in Figure 5, the "noise" introduced by an extra interaction between two individuals in this community could change the shape of the graph, but we should still detect the cycle(using motif) since it is the distinguishing characteristic of this community.

As per the construction of our descriptor, each bin corresponds to presence of a reproducible structure/pattern prevalent in the original graph. Analysis of these bins enables us to learn about different topological information present in multiple classes. In this section, we showcase the sub-structural information we were able to extrapolate that are unique to one class of a dataset.

We compute graphlet-based HOSD descriptors, with a verbosity level of three, for all graphs in two binary bioinformatics datasets; NCI109 [56], and PTC_MR [57]. To ensure simplicity and efficiency, we limit HOSD to a depth of two layers. On the given descriptors, we compare sets of bins across both classes and analyze the ones that have non-zero values in only one class, thereby allowing us to identify sub-structural patterns that only appear in a single class.

In PTC_MR, we observe the interactions between cycles on five and six nodes. After compressing 6-cycles, 5-cycles are only found in class B. The molecules that exhibit this behaviour are show in Figure 4a. Using this, we deduced that for all molecules in class A that include cycles on five and six nodes, 5-cycles are always conjoined to 6-cycles. This exhibits that HOSD can capture how different sub-graphs interact with each other within a graph, and, thereby, across different classes.

We now explore how HOSD captures information about how instances of the same sub-structure interact with each other within a graph. In NCI109, we explore the inter-structural relationships of



(a) Examples from PTC_MR (b) Examples from NCI109

Fig. 4: Molecules from the PTC_MR and NCI109 datasets. The relevant graphlets are shown in black.

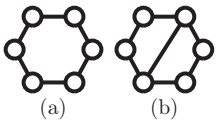


Fig. 5: **(a)** A cycle on six nodes. **(b)** A cycle on six nodes with an extra edge. Note that this graph would not be considered a 6-cycle in terms of graphlets, but would be considered one in terms of motifs.

6-cycles across both classes. In this vein, we explore the higher-order structures found after compressing these cycles. We only take into account the bins that correspond to multiple compressed nodes being found in a sub-graphs. We observe a higher-order pattern only present in one class: 6-cycles that contain multiple compressed 6-cycles are only contained in class A. These are illustrated in Figure

We believe that such information will aid in finding out properties of chemical compounds or networks, and thereby allow the expansion of knowledge in these domains.

4 HOSD-LITE

Finding sub-graphs on large, dense graphs is a computationally expensive and often unfeasible task. With this in mind, we propose the following modifications to the original algorithm:

First, we focus strictly on motifs. We are more likely to detect motifs as opposed to graphlets on more dense graphs, allowing us to compress more nodes and thereby making our new representations smaller. Second, for a given F_i, we perform F_i-compression in a greedy iterative fashion. Once a sub-graph is found, we compress it before looking for more sub-graphs. This will drastically decreasing the search space for our counting procedure. Third, a histogram of degrees is no longer constructed. As per the previous modification, all compressed nodes were originally disjoint sub-graphs, implying that the average degrees for all sub-structures compressed will be the same.

While these changes allow HOSD to scale to larger graphs, we will only be able to compress and encode the information of disjoint instances of an $F_i \in \mathcal{F}$. Moreover, we will obtain different representations based on which sub-graph is picked first, making the descriptor permutation variant. We demonstrate in Section 6 that while this hampers our classification accuracy, it allows HOSD to run on larger datasets.

5 NP-HARDNESS OF FAMILIAL GRAPH COMPRESSION

In this section, we provide some theoretical complexity results regarding Familial Graph Compression (FGC) problem. An instance of FGC comprises of graphs G, H and family of graphs $\mathcal{F}$. First of all we show that FGC is in NP.

Lemma 6. *FGC (G,H,F) is in NP.*

Proof. One can nondeterministically guess :

- the sequence $F_{i_1}, F_{i_2} \ldots, F_{i_{k-1}}$ that results in a sequence of quotient graphs $G = G^0, G^1 \ldots, G^k = H$, and
- a bijection from vertices to G^k to the vertex set of H.

Recall that $k \leq n$. Clearly, this certificate has polynomial size. Further, it can be verified in polynomial time whether $G^j = G^{j-1}/F_{i_{j-1}}$ and whether H and G^k are isomorphic when given a required bijection. Thus FGC is in NP. □

5.1 Hardness of FGC with Motifs

Theorem 7. *FGC is NP-Complete if H is the single vertex graph, and $\mathcal{F}$ has a single motif that is a cycle on n vertices.*

Proof. Lemma 6 shows that FGC is in NP. We prove the hardness by a reduction from Hamiltonian Circuit problem, which is defined as below:

Problem 8. *Given a simple graph $\hat{G}$ on n vertices, does there exist a cycle in $\hat{G}$ that visits every vertex exactly once?*

We will solve FGC with $G = \hat{G}$, H the single node graph, and $\mathcal{F} = \{C_n\}$. If $\hat{G}$ contains a cycle on n vertices then a single compression operation will result single node graph and there are no nontrivial further compressions. On the other hand, if $G = \hat{G}$ doesn't contain a cycle on n vertices then no compressions possible and FGC will return YES in only the trivial case when G is also a single node graph. This we can check manually and report NO. In all other cases, FGC will return NO. This completes the polynomial reduction of Hamiltonian Circuit problem to FGC with the given parameters. The statement follows. □

Theorem 9. *FGC is NP-Complete if G is a simple graph on $n = 3k$ vertices, H is the single vertex graph, and F contains a single motif with exactly k disjoint triangles.*

Proof. Lemma 6 shows that FGC is in NP. We prove the hardness by a reduction from Triangle Packing problem, which is defined as below:

Problem 10. *Given a simple graph $\hat{G} = (\hat{V}, \hat{E})$ on $n = 3k$ vertices, does there exist a partition of $\hat{V}$ into $\hat{V}_1, \hat{V}_2, \ldots, \hat{V}_k$ such that each $\hat{V}_i$ is isomorphic to a triangle in $\hat{G}$?*

We will solve FGC with $G = \hat{G}$, H the single node graph, and $\mathcal{F}$ is a single graph F that is union of k triangles. If $\hat{G}$ contains a partition of $\hat{V}$ into $\hat{V}_1, \hat{V}_2, \ldots, \hat{V}_k$ where each $\hat{V}_i$ is a triangle then a single compression operation will result in a single node graph and there are no further compressions possible. On the other hand, if $G = \hat{G}$ doesn't contain a such a partition of n vertices then no compressions are possible and FGC will return YES in only the trivial case when G is also a single node graph. This can be checked manually and a NO answered to Triangle Packing problem can be reported. In all other cases, FGC will return the desired NO answer. This completes the polynomial reduction as well as the proof. □

5.2 Hardness of FGC with Graphlets

In this context, we recall that the FGC problem is defined as below:

Problem 11 (`Familial Graph Compression` (FGC))**.** *Given a graph G, H, and a family of graphlets $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$, determine whether H can be constructed by an $\mathcal{F}$-compression of G.*

In this section, we will show that FGC is `NP-Hard` by providing a polynomial time reduction from the `EXACTLY-ONE-IN-THREE 3SAT` problem, defined below. Note that the reduction provided is suitable whether the sub-graphs considered are motifs or graphlets, so the choice of sub-graph type is irrelevant to the NP-Hardness of FGC.

Problem 12 (`EXACTLY-ONE-IN-THREE 3SAT`)**.** *Given a boolean formula ϕ in conjuctive normal form, where each clause has exactly three literals, determine whether there exists an assignment of variables such that the every clause in ϕ is satisfiable with exactly one True literal. The variables are from the set $X = \{x_1, x_2, \ldots x_n\}$, and the clauses are from the set $C = \{C_1, C_2, \ldots C_m\}$ where x_{j1}, x_{j2}, x_{j3} are the variables in C_j.*

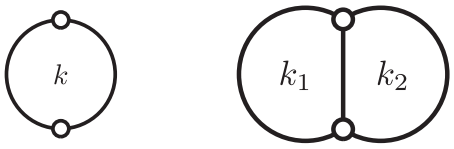
Before we state the result, we must define some important graphs:

Definition 13. *A conjoined cycle, denoted $C_{i,j}$, is a graph with pair of cycles C_i and C_j with exactly one edge common. A node x with degree 3 in a conjoined cycle is called a root.*

Note that there are exactly two roots in a conjoined cycle and they are indistinguishable from each other; thus we call any one of them *the root* of $C_{i,j}$.

Definition 14. *A flower graph, denoted $D_{i,j,k}$, is a graph with three disjoint cycles C_i, C_j, C_k, and a single node r_q that has exactly one neighbor in each of the three cycles, and a separate path P attached to r_q of arbitrary length.*

Theorem 15. *`EXACTLY-ONE-IN-THREE 3SAT` is reducible to the FGC problem in polynomial time.*



(a) Cycle. (b) Conjoined cycles.

Fig. 6: Figure (a) represents a cycle on k nodes. The two explicit vertices are on opposite ends of the cycle. Figure (b) represents two cycles sharing exactly one edge on k_1 and k_2 nodes. The explicit nodes are the only two nodes on which both cycles are incident.

Proof. Let $h(i) := 4i + 1$. Let ϕ be an instance of `EXACTLY-ONE-IN-THREE 3SAT`. We construct an instance of FGC as below:

1) For each variable x_i, we generate graph Z_i as a conjoined cycle $C_{h(i),h(i+2)}$.
2) For each clause C_j containing variables x_{j1}, x_{j2}, x_{j3}, we construct graph S_j has three conjoined cycles $C_{h(j1),h(j1+2)}$, $C_{h(j2),h(j2+2)}$, $C_{h(j3),h(j3+2)}$, with an extra node r_j adjacent to roots of three cycles with three edges, and an extra path P_j of length j attached to r_j.
3) $G = \cup_{1 \le i \le n} Z_i \bigcup \cup_{1 \le j \le m} S_j$
4) Family $\mathcal{F}$ contains all cycles of size 3 to $4n + 2$; additionally there are 3 flower structures for each clause C_j representing each of the possible assignments to literals exactly one of which represents a True literal. For example if all of x_{j1}, x_{j2}, x_{j3} appear nonnegated in C_j, three flowers will be constructed as (1) $C_{h(j1)+1}$, $C_{h(j2)-1}$, $C_{h(j3)-1}$ with roots of each cycle adjacent to r_{j1} (2) $C_{h(j1)-1}$, $C_{h(j1)+1}$, $C_{h(j3)-1}$ with roots of each cycle adjacent to r_{j2}, and (2) $C_{h(j1)-1}$, $C_{h(j2)-1}$, $C_{h(j3)+1}$ with roots of each cycle adjacent to r_{j3}.
5) H is graph on $n + m$ isolated vertices i.e. $E_H = \emptyset$.

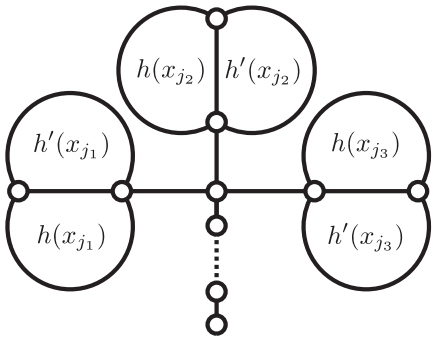
Clearly, the size of construct instance of FGC is polynomial in the size of ϕ. Now, we show why this reduction holds.

Lemma 16. *If ϕ is satisfiable then H_ϕ can constructed by $\mathcal{F}_\phi$-compression of G_ϕ.*

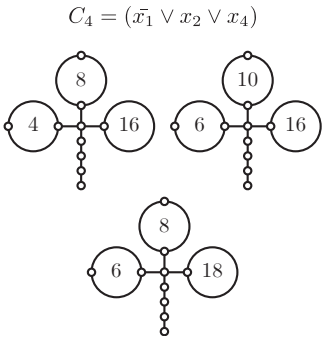
Proof. Given an assignment A of variables that satisfies ϕ, we show how to construct the required sequence a graphs. For each variable x_i set to True, we will perform a C_{4i+1}-compression, and for each variable x_j set to False in satisfying assignment A, we will perform a C_{4j+3}-compression. Following this, we will perform a $D_{i',j',k'}$-compression for all flowers in $\mathcal{F}$ in an arbitrary order. Since assignment A makes exactly one literal true in each clause C_j, corresponding partially-compressed subgraphs S_j will be compressed to a single node by one of three possible flowers in $\mathcal{F}$. Lastly, we perform C_{4i+2}-compression for each x_i set to True and C_{4i}-compression for each x_i set to False. The conjoined cycles Z_i will be compressed to single node as result. We are left with a graph of $n + m$ isolated vertices as required. □

Lemma 17. *If H_ϕ can constructed by $\mathcal{F}_\phi$-compression of G_ϕ then there is a satisfying assignment of variables for ϕ.*

Proof. Let $F_1, F_2, \ldots, F_q$ be a sequence of compressions that gives a graph with $n + m$ isolated vertices, clearly S_i's can only be compressed by one of the 3 valid graphlets. An arbitrary



(a) Flower graph of a clause.



(b) Family corresponding to a clause.

Fig. 7: The sub-graph in G corresponding to clause C_j is presented in (c), where $h'(x_i) = h(x_i) + 2$. Four examples of members of $\mathcal{F}$ corresponding to the clause written above are provided in (d).

Z_j can compressed in two different ways:(1) C_{4i+1}-compression followed by a C_{4i+2}-compression, or (2) C_{4i+3}-compression followed by a C_{4i}-compression. If the valid compression sequence contains a C_{4i+1} graph, we set x_j to True otherwise x_j is set to False. Clearly, this is a valid satisfying assignment for CNF ϕ. □

This concludes the proof of theorem. □

We observe that a few trivial changes to the above reduction also give the following result - we leave the details for the reader to establish.

Theorem 18. *FGC remains* `NP-Hard` *for an instance where G is a connected graph and H is a single vertex graph.*

6 Experiments and Discussion

TABLE 1: The number of graphs, maximum number of nodes in a graph, maximum number of edges in a graph, maximum degree of a node throughout all graphs, and number of classes for each dataset used in our experimental analysis.

Dataset	$\|G\|$	N_{max}	E_{max}	D_{max}	$\|m\|$
MUTAG	188	28	33	4	2
NCI1	4110	111	119	4	2
NCI109	4127	111	119	5	2
PTC	344	64	71	4	2
PROTEINS	1113	620	1049	25	2
ENZYMES	600	125	149	9	6
DD	1178	5748	14267	19	2
IMDB-BINARY	1000	136	1249	135	2
IMDB-MULTI	1500	89	1467	88	3
REDDIT-BINARY	2000	3782	4071	3062	2
REDDIT-MULTI	4999	3648	4783	2011	5

Due to the interpretability of our descriptors, it is logical to work with interpretable machine learning paradigms. With this in mind, we opted for the random forest classifier on a multitude of datasets and compared it to the current state of the art graph descriptors.

Experimental Setup All experiments were conducted on an Intel-Core i7-4700 processor with 8 logical cores and 8GB RAM. We tested HOSD and HOSD-Lite on all three verbosity levels. HOSD is evaluated on both graphlets and motifs. After obtaining representations from all these methods, results were assessed on the best parameters through 10-fold cross-validation. We repeated this assessment 10 times, and here report the mean best results.

Datasets We experimented on seven bioinformatics datasets: MUTAG, NCI1, NCI109, PTC, PROTEINS (PROT), ENZYMES (ENZ), and DD; and four social datasets: IMDB-BINARY (IMDB-B), IMDB-MULTI (IMDB-M), REDDIT-BINARY (RDT-B), and REDDIT-MULTI-5K (RDT-M) [24].

Hyperparamter Tuning For a d-dimension feature vector $\sqrt{d}$ features were chosen to build a tree. The range of numbers of trees are chosen from the set $\{50, 100, 500\}$. Gini impurity was used as a metric to build the tree and no constraint was imposed on the size of leaf nodes. Samples were bootstrapped and the number of samples to make a split in tree was chosen from the set $\{2, 3, 4, 5\}$.

HOSD Parameters We chose the family $\mathcal{F}$ to include all cycles, cliques, paths, and stars on three to six nodes. While evaluating HOSD on larger datasets (Proteins, DD, and IMDb), we used a similar family up to four nodes, giving us a total of six graphs. The family for REDDIT was only up to three nodes. This constraint was unnecessary on HOSD-Lite due to its scalability. To ensure efficiency, we limited HOSD to a depth of two layers.

Classification Results The accuracies on each dataset obtained for each setting of HOSD and HOSD-Lite is presented in the supplementary document. The best results are obtained by graphlet-based HOSD, with verbosity 3. Similarly, HOSD-Lite achieved the best results with verbosity 3.

In Table 3 we compare the best HOSD and HOSD-Lite method to three other graph descriptors: FGSD [47], NetSimilie [42], and NetLSD (LSD) [46]. We set the bin-width 0.0001 for FGSD, as recommended by Tsitsulin et al. [46]. We use all the variant of NetLSD mentioned in their work, and utilized the entire eigenspectrum. The best of these results are reported.

Observe that HOSD outperforms most of them. Even for the datasets where our proposed models fall short, HOSD or HOSD-Lite's accuracy remains still within 1% of the best. Superior classification accuracy provides evidence that HOSD is able to preserve distinguishing graph features by using simple graph-theoretic statistics at multiple scales, while still maintaining interpretability.

In Table 4 we compare our method to the WL-subtree Kernel (WL) [16] – a standard baseline in concurrent graph classification literature. For this comparison, we borrowed the values provided by Xu et al. from their work on the Graph Isomorphism Network [33], as they followed a similar experimental setup. Observe that the results provided by HOSD are competitive even to the WL-Kernel.

TABLE 2: Details of classification accuracies of HOSD and HOSD-Lite with Graphlet (G) and Motif (M) use with varying verbosity levels.

Dataset	G_1	G_2	G_3	M_1	M_2	M_3	M_{L1}	M_{L2}	M_{L3}
MUTAG	85.75	85.78	86.15	85.69	85.71	86.04	87.87	87.47	87.20
NCI1	78.54	79.59	80.31	78.33	79.52	80.02	71.83	72.11	67.96
NCI109	77.33	78.85	79.00	77.39	78.88	79.05	70.52	70.92	66.68
PTC_MR	60.18	60.21	61.06	59.92	59.96	60.78	59.46	59.13	59.37
PROTEINS	72.23	72.33	72.28	72.48	72.63	72.31	74.45	74.42	73.99
ENZYMES	49.07	49.47	49.87	47.45	47.42	47.30	36.20	36.23	33.80
DD	78.99	78.63	78.19	78.50	78.73	77.81	77.10	76.98	76.68
IMDB-BINARY	73.17	73.42	73.47	72.98	72.93	73.03	70.85	70.78	69.26
IMDB-MULTI	49.98	50.04	49.98	48.77	48.89	48.68	47.94	47.75	46.61
REDDIT-BINARY	89.52	89.82	89.53	89.76	89.96	89.66	89.89	89.46	89.35
REDDIT-MULTI-5K	53.75	54.12	53.98	53.93	54.25	54.12	52.41	51.75	51.13

TABLE 3: Comparison with other state of the art graph embeddings. The bold numbers show the results which lie within 1% of the best. OME stands for out-of-memory error.

Dataset	FGSD	NetLSD	NetSIMILIE	$HOSD_{G3}$	HOSD-$Lite_{M1}$
MUTAG	**88.07**	85.28	84.63	86.15	**87.87**
NCI1	**79.60**	76.31	73.96	**80.31**	71.83
NCI109	**79.47**	76.12	72.89	**79.00**	70.52
PTC_MR	58.91	**61.19**	59.09	**61.06**	59.46
PROTEINS	70.18	**74.63**	71.15	72.28	**74.45**
ENZYMES	33.95	44.25	42.10	**49.87**	36.20
DD	76.60	**77.27**	74.72	**78.19**	77.10
IMDB-BINARY	**73.88**	**73.79**	**74.41**	**73.47**	70.85
IMDB-MULTI	**50.55**	**50.57**	49.32	**49.98**	47.94
REDDIT-BINARY	81.60	89.12	**89.53**	**89.53**	**89.89**
REDDIT-MULTI-5K	OME	**53.58**	52.78	**53.98**	52.41

TABLE 4: Comparison with state of the art classification paradigms. The bold numbers show the results which lie within 1% of the best.

Dataset	WL	$HOSD_{G3}$	HOSD-$Lite_{M1}$
MUTAG	**90.40**	86.15	87.87
NCI1	**86.00**	80.31	71.83
PTC_MR	59.90	**61.06**	59.46
PROTEINS	**75.00**	72.28	**74.45**
IMDB-BINARY	**73.80**	**73.47**	70.85
IMDB-MULTI	**50.90**	**49.98**	47.94
REDDIT-BINARY	81.00	**89.53**	**89.89**
REDDIT-MULTI-5K	52.50	**53.98**	52.41

We choose not to compare to Graph Neural Networks, as the comparison is unfair; GNNs are steered towards finding the "right" patterns via the minimization of classification loss. In contrast, we produced vector representations without this goal being deliberately ingrained within our proposed model.

7 Conclusion

This work presents the Higher Order Structure Descriptor: an interpretable, permutation invariant graph descriptor that captures the structural information of a graph at multiple scales. We show how a simple sub-graph compression technique, coupled with relevant structural knowledge, allows us to encapsulate rich information pertaining to a graph. Using a family of 14 special graphs and only two layers, we demonstrate that HOSD performs well on benchmark unlabeled graph datasets, and offers results competitive with recently introduced state-of-the-art graph descriptors. Our future work involves using data-driven approaches to choosing an appropriate family, and catering to the running time that affects its applicability. We also plan on generalizing HOSD to work with labeled graphs. This can be done trivially by concatenating metadata relevant to the graph onto the descriptor produced by HOSD. However, we wish to explore a more structured approach that involves including labelled graphs in the family, allowing us to capture more coarse and domain-driven information.

Acknowledgments

The authors would like to thank...

References

[1] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.

[2] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl.-Based Syst.*, vol. 151, pp. 78–94, 2018.

[3] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 1616–1637, 2018.

[4] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[5] R. Read and D. Corneil, "The graph isomorphism disease," *Journal of Graph Theory*, vol. 1, pp. 339–363, 1977.

[6] R. Kondor and H. Pan, "The multiscale laplacian graph kernel," in *Advances in Neural Information Processing Systems*, 2016, pp. 2990–2998.

[7] D. Haussler, "Convolution kernels on discrete structures," Technical report, Department of Computer Science, University of California , Tech. Rep., 1999.

[8] 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.

[9] H. Wiener, "Structural determination of paraffin boiling points," *Journal of the American Chemical Society*, vol. 69, no. 1, pp. 17–20, 1947.

[10] J. Ramon and T. Gärtner, "Expressivity versus efficiency of graph kernels," in *First international workshop on mining graphs, trees and sequences*. Citeseer, 2003, pp. 65–74.

[11] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2006, pp. 163–172.

[12] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, 2009, pp. 488–495.

[13] K. M. Borgwardt, T. Petri, S. Vishwanathan, and H.-P. Kriegel, "An efficient sampling scheme for comparison of large graphs," *Mining and Learning with Graphs, MLG*, pp. 1–3, 2007.

[14] 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.

[15] N. Shervashidze and K. Borgwardt, "Fast subtree kernels on graphs," in *Advances in neural information processing systems*, 2009, pp. 1660–1668.

[16] 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.

[17] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[18] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K. Borgwardt, "Scalable kernels for graphs with continuous attributes," in *Advances in Neural Information Processing Systems*, 2013, pp. 216–224.

[19] C. Morris, K. Kersting, and P. Mutzel, "Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 327–336.

[20] M. Neumann, N. Patricia, R. Garnett, and K. Kersting, "Efficient graph kernels by randomization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 378–393.

[21] L. Bai and E. R. Hancock, "Fast depth-based subgraph kernels for unattributed graphs," *Pattern Recognition*, vol. 50, pp. 233–245, 2016.

[22] F. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya, "Global graph kernels using geometric embeddings," in *Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014.

[23] L. Bai, L. Rossi, H. Bunke, and E. R. Hancock, "Attributed graph kernels using the jensen-tsallis q-differences," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 99–114.

[24] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1365–1374.

[25] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, "Propagation kernels: efficient graph kernels from propagated information," *Machine Learning*, vol. 102, no. 2, pp. 209–245, 2016.

[26] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[27] B. Gaüzere, L. Brun, and D. Villemin, "Two new graphs kernels in chemoinformatics," *Pattern Recognition Letters*, vol. 33, no. 15, pp. 2038–2047, 2012.

[28] G. Nikolentzos and M. Vazirgiannis, "Enhancing graph kernels via successive embeddings," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 1583–1586.

[29] G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, "Kernel graph convolutional neural networks," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer International Publishing, 2018, pp. 22–32.

[30] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[31] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[32] D. V. Tran, N. Navarin, and A. Sperduti, "On filter size in graph convolutional networks," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1534–1541.

[33] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[34] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1666–1674.

[35] 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, 2019, pp. 4602–4609.

[36] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[38] ——, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[39] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

[40] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[41] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[42] 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*. ACM, 2013, pp. 1439–1440.

[43] R. Kondor, N. Shervashidze, and K. M. Borgwardt, "The graphlet spectrum," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 529–536.

[44] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Graph classification via topological and label attributes," in *Proceedings of the 9th international workshop on mining and learning with graphs (MLG), San Diego, USA*, vol. 2, 2011.

[45] A. Dutta and H. Sahbi, "Stochastic graphlet embedding," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2018.

[46] A. Tsitsulin, D. Mottin, P. Karras, A. M. Bronstein, and E. Müller, "Netlsd: Hearing the shape of a graph," in *KDD*, 2018.

[47] S. Verma and Z.-L. Zhang, "Hunt for the unique, stable, sparse and fast feature learning on graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 88–98.

[48] D. Marcus and Y. Shavitt, "Rage–a rapid graphlet enumerator for large networks," *Computer Networks*, vol. 56, no. 2, pp. 810–819, 2012.

[49] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, vol. 30, no. 4, pp. 559–565, 2014.

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

[51] M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. Borgwardt, "Near-optimal supervised feature selection among frequent subgraphs," in *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, 2009, pp. 1076–1087.

[52] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *2015 IEEE International Conference on Data Mining*. IEEE, 2015, pp. 1–10.

[53] N. K. Ahmed, J. Neville, R. A. Rossi, N. G. Duffield, and T. L. Willke, "Graphlet decomposition: Framework, algorithms, and applications," *Knowledge and Information Systems*, vol. 50, no. 3, pp. 689–722, 2017.

[54] M. C. Golumbic and I. B.-A. Hartman, *Graph theory, combinatorics and algorithms: Interdisciplinary applications*. Springer Science & Business Media, 2006, vol. 34.

[55] R. Uehara, S. Toda, and T. Nagoya, "Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs," *Discrete Applied Mathematics*, vol. 145, pp. 479–482, 2005.

[56] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008.

[57] C. Helma, R. D. King, S. Kramer, and A. Srinivasan, "The predictive toxicology challenge 2000–2001," *Bioinformatics*, vol. 17, no. 1, pp. 107–108, 2001.

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.