Interpretable Multi-scale Graph Descriptors via Structural Compression

Zohair Raza Hassan[*][**], Ammar Ahmed[*], and Mudassir Shabbir

Department of Computer Science, Information Technology University of the Punjab, Pakistan
{zohair.raza, ammar.ahmed}@itu.edu.pk, mudassir@rutgers.edu

Abstract. Graphs can efficiently model a wide range of real-world structures that involve relations between pairwise entities such as chemical compounds, road networks, internet traffic, etc. Designing concise graph representations which preserve relevant topological information would allow the use of rich machine learning toolsets for the analysis, classification, and clustering of graph structured data. Some notable graph representations based on spectral features, graph-theoretic properties, and data-driven information have been proposed. While fairly fruitful in the aforementioned applications, most of these representations either lack interpretability or are unable to effectively encode a graph's structure at both a local and global scale. In lieu of this, we propose the Higher-Order Structure Descriptor (HOSD): an interpretable graph descriptor that captures information about the sub-structures found in a graph at multiple scales. HOSD is easy to implement, straightforwardly interpretable, and invariant to node permutations due to its graph-theoretic nature. A faster version, HOSD-Lite, is also presented to approximate HOSD on denser graphs. HOSD and HOSD-Lite are evaluated on multiple real-world data sets 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.

Keywords: Graphs · Hierarchical representations · Classification.

1 Introduction

Due to their discrete nature and effectiveness in representing relationships between pairs of entities, graphs can be used to model a plethora of practical structures such as molecules [6], social networks [2], and airline routes [22]. 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. [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 three main challenges in transferring existing machine learning technology to graphs:

[*] Authors contributed equally to this work
[**] Corresponding author

– Graphs in a data set corresponding to a problem may have arbitrarily different sizes, both in terms of vertices and edges. This is a problem because traditional classifiers expects 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.
– Neighborhood structure of vertices in graph is much more complex in shape and as such can't be modelled by tensors. Attempts to approximate this structure to low dimension vectors have not proved fruitful so far.
– In many examples, essential information regarding a network lies with presence or the absence of edges and vertices themselves do not carry any labels. In such cases of unlabelled graphs, many seemingly different looking networks are, in essence, copies of the same graph with different orders on the vertex set. This is the classical graph isomophism problem:

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

  Though not known to be NP Hard, the notorious GI problem has evaded a polynomial algorithm

Graph kernels [16], vector descriptors [29, 30], and, more recently, graph convolutional networks (GCNs) [10] have been proposed to solve one or more of these issues. GCN's and graph kernel methods have reported good classification accuracies but these techniques do not scale well, and are not informative enough with respect to the patterns learnt to discern among categories of graphs. Recently introduced state-of-the-art graph descriptors are based on the spectral features of a graph's Laplacian matrix $L = D - A$ where D is diagonal matrix with $D_{i,i}$ is the degree of vertex i and A is the adjacency matrix of the graph [29, 30]. Despite providing excellent results, these descriptors do not provide an intuitive reasoning as to why a graph may belong to a certain category. On the other hand, a graph theoretic approach that looks at sub-structures would be more informative; one can observe what kind of structures are being used to make a decision.

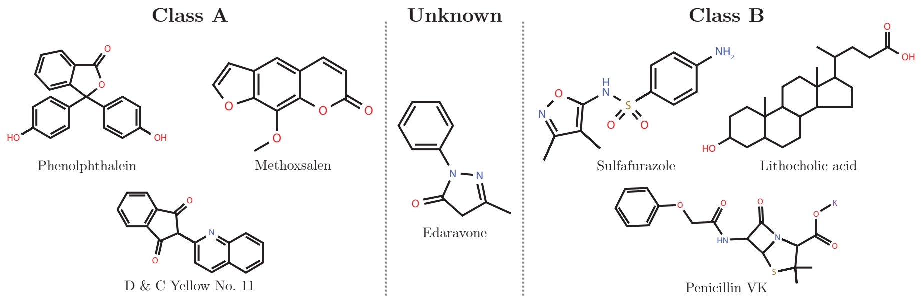

Fig. 1. Chemical compounds taken from the Predictive Toxicology Challenge (Male Rats).

Consider chemical data sets for example: it is more informative for a chemist to know what kind of molecular structure is observed to classify a chemical, as opposed to what kind of pattern was observed in the eigen-spectrum of it's Laplacian. To illustrate, take the polycyclic compounds given in Figure 1. Considering only the topological structure of these molecules, we learn that only graphs belonging to class B have disjoint instances of rings with five and six atoms. Using this information, not only do we correctly infer that the unknown graph belongs to Class B, we also learn an interpretable, topological feature prevalent in the dataset that distinguishes molecules of both classes.

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, broadly, be divided into three categories as below.

2.1 Graph Convolutional Neural Networks

Highly cited paper should be mentioned/included in our related work [25].

Inspired by the success of convolutional neural networks in image processing and computer vision, Graph CNNs learn to classify data via convolutions over the graph or their Laplacian. In these works, each graph is treated as a signal on which convolutional filters and pooling has been applied for feature extraction [13, 17]. In [4] 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. In an other work, using convolutional filters, fingerprints of graphs are calculated by hashing neighborhood information and node labels [15].

See [11] for detailed study of which include deep-learning based method for graph classification. These approaches rely on improving the pipeline of neural network with innovative operators and filters that work well with graphs. Most use edge and node labels, as these 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.

Why do we talk about embeddings here?

2.2 Graph Kernels

Graph kernels are R-convolution kernels 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

Add a reference to Weiner Indices. 1947

Ramon and Gaertner, 2004

Neuhaus and Bunke, 2006

Also cite B., Petri, et al., MLG 2007

I think we should remove this line.

Tell me about this paper - what do they do?

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 [31]. 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 [26] , edit-distance [24] .

The Graphlet Kernel presented in [28] compares 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. These solutions will miss structures in graphs on a larger scale [9]. The Weisfeiler-Lehman Kernel [27] based on Weisfeiler-Lehman graph isomorphism test.

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 [18],which builds a hierarchy of nested sub-graphs.

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 [7], or the presence of specific sub-structures. Dutta et al. [14] propose using random walks coupled with hashing to capture the presence of different sub-graphs in a graph. However, these approaches are unable to capture structural information about the graph at higher scales. Tsitsulin et al. [29] and Verma et al. [30] 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 categories of graphs.

3 Higher-Order Structure Descriptor

Scale-space theory is built upon the idea that multi-scale representation of data is imperative to capturing the structure of "unknown real-world signals" [20]. A canonical instance of such multi-scale representations was the advent of image pyramids in computer vision; a hierarchical representation for images was achieved via successively down-sampling an image after applying a Gaussian filter to it [1]. The construction of these representations allows us to look for patterns on multiple scales. This multi-scale approach has proven to be successful, bearing fruit in the construction of scale invariant feature extraction paradigms for instance [5, 21]. Inspired by this success, we propose following a similar methodology on graphs.

Translating this process directly is not a straightforward task: graphs are discrete structures which only encode pairwise relationships, whereas images are grid based structures which embody spatial correspondences between each pixel. Recall that a Gaussian filter is a low-pass filter that is able to compress the information of multiple adjacent pixels into one. Analogously, one could compress multiple nodes of a graph into one super-node. The creation of multiple such super-nodes would provide us with a new higher-order representation of our graph.

Since graphs embody structural information, we choose to compress nodes based on such information. Hence, we proceed by choosing a sub-graph and compressing together nodes which are present in such sub-graphs. More precisely, each non-disjoint instance of these sub-graphs is compressed into super-nodes. An added advantage of the use of sub-graphs is that it allows the capturing of structural correspondence between these sub-structures as we progress to higher scale representations. In their work on the Multi-scale Laplacian Kernel, Kondor et al. [18]give the example of a molecular graph, stating that local features could tell us about the functional groups present in the molecule, but global features encode where their groups lie within the graph.

Realistically speaking, a graph may have many prevalent sub-structures that may be compressed to reveal important information. Thereby, in lieu of using one sub-graph, we propose making multiple representations by using a set of sub-graphs to construct the hierarchical representations of G.

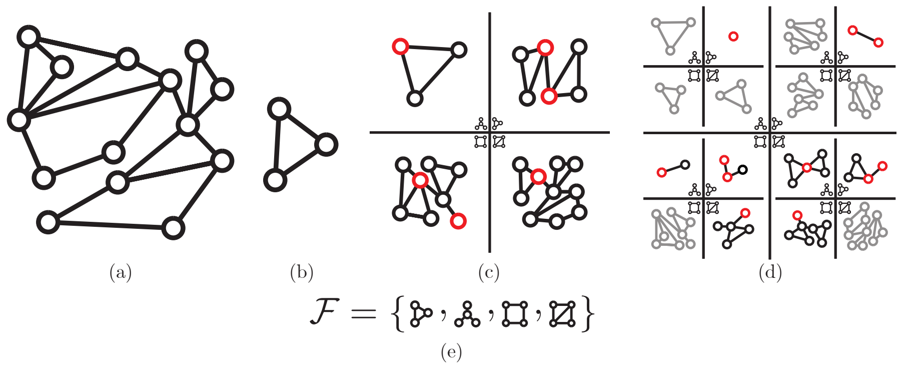

Fig. 2. (a) The input graph, G. **(b)** The target graph, H. **(c)** Second layer higher-order representations of G obtained after independently compressing each sub-graph in the family $\mathcal{F}$. New compressed nodes are colored red. **(d)** Third layer higher-order representations of G obtained after compressing sub-graphs from the representations of G obtained in (b). Note that when instances of a sub-graph are not found in G, the resultant graph is ignored. This is highlighted by presenting the newer representations with a lower opacity. New compressed nodes are colored red. **(e)** The family, $\mathcal{F}$, of sub-graphs being compressed.

3.1 Preliminaries

Let $\mathcal{D} = \{G_1, G_2, \dots\}$ be a database of graphs: $G_i = (V_i, E_i)$, where V_i is a set of entities and E_i represents pairwise interactions between these entities. Each G_i is undirected and unweighted, i.e. for $u, v \in V_i$, an edge (u, v) is same as the edge (v, u), and these links are binary in nature. The graphs in $\mathcal{D}$ are of variable sizes, that is, the number of nodes and number of edges may not match in different networks. 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. We further assume the existence of function $f : D \rightarrow \{1, 2, \ldots, m\}$ i.e there is a unique class associated with each graph. The following is the definition of the general classification problem on graphs.

Problem 2 (Graph Classification). Given a finite sample $\mathcal{D}' \subseteq \mathcal{D}$ of a database of graphs that belong a set of m classes, generate a estimation f' such that f' agrees with f on maximum number of graphs in $\mathcal{D}$.

Since $\mathcal{D}$ can be infinite in theory, it makes sense to evaluate f' on a uniform sample of $\mathcal{D}$. However, as discussed earlier, learning such a function on graphs poses serious challenges due to the permutation and size invariance of graphs in a given database. Thus, it makes sense to instead learn a on an embedding of graphs and has recently been a promising line of work.

There are two definitions of sub-graphs popular in literature: graphlets, and motifs. Let $G = (V, E)$ and $F = (V_F, E_F)$ be two graphs. Let $G' = (V', E')$ be a sub-graph of G such that $V' \subseteq V$ and $E' = \{(u, v) : u, v \in V' \wedge (u, v) \in E\}$. F is a graphlet in G, incident on the nodes V', iff. there exists a function $\phi : V' \rightarrow V_F$, such that $E_F = \{(\phi(u), \phi(v)) : (u, v) \in E'\}$. F is a motif in G, incident on the nodes V', iff. there exists a function $\phi : V' \rightarrow V_F$ such that $E_F \subseteq \{(\phi(u), \phi(v)) : (u, v) \in E'\}$.

Let $l \in \{1, 2, 3, ...\}$ be an input parameter that dictates the maximum depth of our model i.e. how many times G is successively compressed. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ be a family of graphs. Clearly, if the family $\mathcal{F}$ is chosen suitably enough, a histogram representing $\mathcal{F}$ contains all the information about the graph G, has a fixed linear ordering, and is unique for all isomorphisms of G. Trivially, a family that contains all graphs in a given database will give us a unique vector for each $G_i \in \mathcal{D}$. However, such a representation would not be informative enough to perform classification. Thus, the choice of graphs in the family $\mathcal{F}$ is the most important part of this problem.

3.2 Algorithm

Now, we can begin to formally describe our model, the Higher-Order Structure Descriptor (HOSD). In this section, we describe how HOSD constructs a descriptor for a single graph, G. For a dataset of graphs, the algorithm is simply repeated for each $G_i \in \mathcal{D}$. We begin with a simple sketch of the algorithm before describing each step in detail in the subsections to follow. For a single graph, the time complexity of each step is discussed in subsection before analysing the overall time complexity in Section 3.6. In this section, for the sake of simplicity, the term representation is may refer to the original input graph, or a graph obtained after compressing nodes in the original graph. The final vector output by HOSD is referred to as a descriptor.

Sketch Given G, $\mathcal{F}$, and l, HOSD constructs a histogram that encodes the spatial structure of G based on $\mathcal{F}$, at l different scales. We begin by finding all instances of $F_i \in \mathcal{F}$

within G. We construct a descriptor for G that includes $|V|$, $|E|$, and a histogram of the number of instances of each $F_i \in \mathcal{F}$. Note that the algorithm would terminate here if $l = 1$. For each $F_i \in \mathcal{F}$, $\mathcal{F}$ new representations of G are obtained by compressing the instances of F_i found. As earlier, a histogram is constructed for these new representations as well. If $l > 2$, we compress each representation in a similar manner to get $|\mathcal{F}|^2$ more representations. In general, a total of $\sum_{j=0}^{l-1} |\mathcal{F}|^j$ representations are obtained, and histograms are constructed for each representation. In the end, these histograms are simply concatenated to construct our descriptor for G.

Finding instances of $\boldsymbol{F_i \in \mathcal{F}}$ For ease of implementation, we make use of the VF2 algorithm described by Cordella et al., which takes $O(|V|! \times |V|)$ time in the worst case [12]. However, this time complexity is not representative of our model; any sub-graph finding method that returns node mappings may be employed in HOSD, and time complexity varies greatly depending on the type of graphs in $\mathcal{F}$. For instance, a trivial algorithm for finding k-cliques in G would take $O(\binom{|V|}{k} \times k^2)$.

One aspect to consider when detecting sub-graphs is whether one should consider graphlets or motifs. Not wanting to limit our model to a seemingly arbitrary choice, we implement and evaluate HOSD by considering both types of sub-graphs in Section 4.

Compressing sub-graphs of $\boldsymbol{G}$ Assume that for a representation, G, and $F_i \in \mathcal{F}$ we have a set $M = \{M_1, M_2, \dots : M_j \subseteq V\}$ that includes the nodes in G which map to an instance of F_i. There may be instances of F_i where nodes are shared, i.e. $M_j \cup M_k \neq \emptyset, j \neq k$. One may choose to make different representations for each disjoint instance, or randomly choose disjoint instances of M_i to compress. However, to keep our model simple and deterministic, we compress these instances that share nodes together. To do so, a new graph is constructed: $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V} = \bigcup_{j:M_j \in M} M_j$ and $\hat{E} = \bigcup_{j:M_j \in M} E'_{M_j}$, where $E'_{M_j} = \{(u,v) : u, v \in M_j \wedge (u,v) \in \hat{E}\}$. Then, each connected component is identified in $O(|\hat{V}| + |\hat{E}|)$ time via breadth-first-search [19]. Finally, each component is compressed into a single node.

It can be observed in Figure 4 that this leads to infinitely many different graphs having the same higher-order representation. For HOSD to differentiate between these different graphs, it must be able encode the spatial structure amongst the non-disjoint instances of F_i. To do so, a histogram of the average degrees of each connected component in $\hat{G}$ is constructed and concatenated to the final descriptor. Since real numbers are obtained during this averaging, the bin-width to 0.1 to keep the size of the final histogram small. The limits of the histogram of degrees is decided by the minimum and maximum average degrees observed throughout the entire dataset $\mathcal{D}$. Computing the average degrees theoretically takes $O(|\hat{V}|)$ time. Refer to Figure 3 for an illustration of this step.

Let $\bar{G} = (\bar{V}, \bar{E})$ be the representation received after the commpression has taken place. To keep track of which level a node was contracted on, each compressed super-node $v \in \bar{V}$ is assigned a label, α_v, equivalent to the layer it was compressed on i.e. $\alpha_v = t$ for layer t. In the original input graph, $\alpha_v = 1$ for all each node.

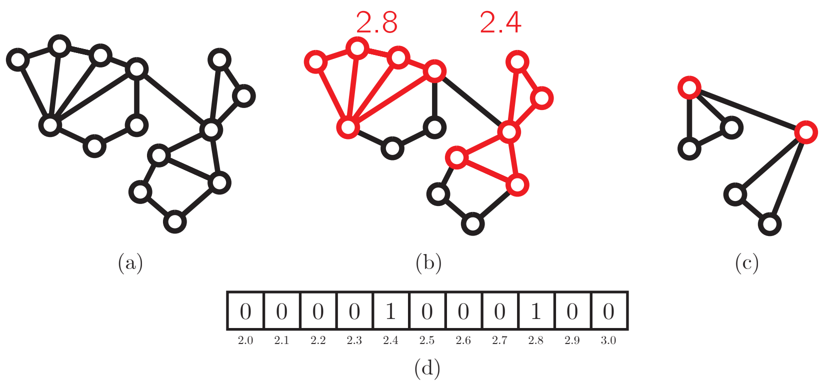

Fig. 3. (a) The input graph, G. **(b)** All instances of triangles in G, highlighted in red, alongside the average degrees of each disjoint sub-structure. **(c)** The graph obtained after compressing each disjoint sub-structure highlighted in (b). **(d)** The resultant histogram of degrees

Note that, in the scenario where no instance of F_i is found to compress i.e. $M = \emptyset$, bins corresponding to the information of the graph received post compression are set to zero.

Counting instances of $\boldsymbol{F_i} \in \boldsymbol{\mathcal{F}}$ For each representation, once all instances of F_i are found, there are a number of ways one can store said information in the descriptor due to the labels we assign during compression. Incorporating these features would allow us to capture richer information about how the sub-structures of G propagate as we go deeper into our layers. 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:

- **Verbosity 1:** Ignore α_v's completely and keep one bin for each $F_i \in \mathcal{F}$, giving us $|\mathcal{F}|$ individual bins.
- **Verbosity 2:** For each super-node, create a separate bin for different combinations of each α_v possible. Thereby, on the t^{th} layer, for each $F_i \in \mathcal{F}$, there are a total of $|\mathcal{F}| \times 2^{t-1}$ bins for each representation.
- **Verbosity 3:** Similar to verbosity 2, except separate bins are also kept for the number of compressed nodes found in a sub-structure i.e. there are separate bins depending on the number of $v \in V$ found such that $\alpha_v \neq 1$. This gives us a total of $(\beta+1) \times |\mathcal{F}| \times 2^{t-1}$ bins for each representation, where β is the maximum number of nodes for a graph $F_i \in \mathcal{F}$.

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 4.

3.3 Choosing a family $\mathcal{F}$

As we previously observed, the choice of family will directly affect the kind of information that is encoded in our histogram. In congruence with our graph-theoretic approach, we propound that a suitable family is one that is able to completely compress each graph in our dataset down to one node. This family must also be non-trivial; trivial families that allow complete compression G are ones that include G or just an edge if G has only one connected component. Much to our dismay, finding such a family for even one graph proves to be a daunting task; in Appendix A, we show that, in the general case, given a graph G, and a family $\mathcal{F}$, it is NP-Hard to determine whether $\mathcal{F}$ family can completely compress G. Keeping this constraint in mind, it would be wise to choose families based on domain knowledge for the problem at hand. For instance, for chemical classification problems, including a cycle on six nodes would help encode information about benzene rings.

Note, as HOSD propagates through each layer, we are able to encode the presence of larger sub-structures - granted the use of the right verbosity level. These larger sub-structures are essentially conjoined combinations of the structures present in $\mathcal{F}$, as illustrated in Figure 4 using a 4-star and triangle. Keeping this in mind, it would be wise to avoid including sub-graphs that could be built by combining other sub-graphs already in $\mathcal{F}$.

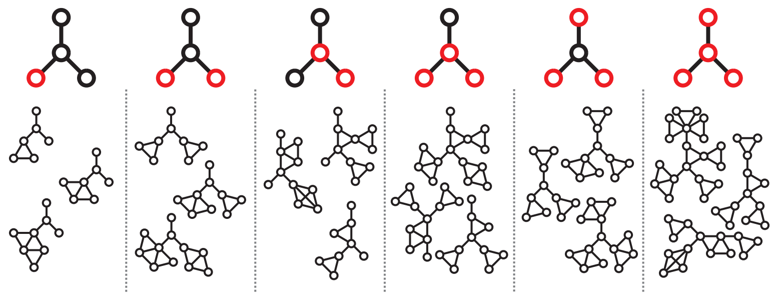

Fig. 4. Possible sub-structures that HOSD (Verbosity 2 and 3) can identify by finding instances of 4-stars nodes after compressing triangles.

3.4 Interpretting HOSD

As per the construction of our descriptor, each bin corresponds to presence of a reproducible structure/pattern prevalent in the original graph. HOSD is especially informative about the the sub-structures prevalent within a graph, and analysis of the relevant bins enables us to learn about different topological structures present in multiple classes.

Using any machine learning paradigm that returns the importance of features, such as decision trees or random forests, one can find out the features that helped distinguished between multiple classes of graphs. For instance, consider again the example in Figure 1. 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.

3.5 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:

– 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.
– For a given F_i, once a sub-graph is found, compress it before looking for more sub-graphs. Essentially, we are ignoring the nodes we have already found an instance of F_i on, drastically decreasing the search space for the rest of our counting procedure.
– 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 4 that while this hampers our classification accuracy, it allows HOSD to run on larger datasets.

3.6 Time Complexity

Let ξ_i be the maximum time taken to find a sub-graph F_i from $\mathcal{F}$ in $G = (V, E)$. As described in the previous sections, the maximum running time on one graph is $O(|\mathcal{F}| \times (|V| + |E|) + \sum_{i:F_i \in \mathcal{F}} \xi_i)$. In the worst case, $O((\beta+1)|\mathcal{F}|2^l)$ bins are updated in the histogram. This process is repeated for $O(|\mathcal{F}|^l)$ representations for said graph, giving us the following time complexity for one graph:

$$O(|\mathcal{F}|^l(|\mathcal{F}|(|V| + |E|) + \sum_{i:F_i \in \mathcal{F}} \xi_i + (\beta+1)|\mathcal{F}|2^l)) \tag{1}$$

4 Experiments and Discussion

We evaluate HOSD on a multitude of datasets prevalent in graph classification literature. This includes seven bio-informatics datasets, and four social network datasets. Due to the interpretability of our descriptors, it is logical to work with interpretable machine learning paradigms. With this in mind, we opt for the random forest classifier (Graph Higher Order Histogram) on a multitude of datasets and compare it to the current state of the art graph classification paradigms.

	$\|G\|$	N_{avg}	N_{max}	N_{min}	E_{avg}	E_{max}	E_{min}	$avg(D_{avg})$	D_{max}	$\|m\|$
MUTAG	188	17.93	28	10	19.79	33	10	2.189	4	2
NCI1	4110	29.87	111	3	32.30	119	2	2.163	4	2
NCI109	4127	29.68	111	4	32.13	119	3	2.165	5	2
PTC	344	14.29	64	2	14.69	71	1	1.98	4	2
PROTEINS	1113	39.06	620	4	72.82	1049	5	3.735	25	2
ENZYMES	600	32.63	125	2	62.14	149	1	3.866	9	6
DD	1178	284.32	5748	30	715.66	14267	63	4.8	19	2
IMDB-BINARY	1000	19.77	136	12	96.53	1249	26	8.89	135	2
IMDB-MULTI	1500	13.00	89	7	65.94	1467	12	8.10	88	3
REDDIT-BINARY	2000	429.63	3782	6	497.75	4071	4	2.338	3062	2
REDDIT-MULTI	4999	508.52	3648	22	594.87	4783	21	2.25	2011	5

Table 1. Basic stats of datasets used for classification, N, E, D, m denotes nodes, edges, degree and classes respectively

4.1 Experimental Setup

All experiments are conducted on an Intel-Core i7-4700 processor having 8 logical cores with 8GB RAM. Here we compare our methods with FGSD [30], NetLSD [29], NETSIMILE [7] and graph2vec [23]. . Each method is required to be completed within 24 hours otherwise it should be terminated. We set the bin-width 0.0001 for FGSD, as recommended by Tsitsulin [29]. We use all the hyper-parameters of NetLSD as mentioned in their work, and utilized all the eigenvalues. The best of these results are reported. A representation of a graph with dimentions 35 is obtained using NETSIMILIE to compare their results. We obtained representation of graphs using default hyper-parameteres of graph2vec. After obtaining representations from all these methods, results are assessed on the best parameters through 10-fold cross-validation. We repeated this assessment 10 times, and here report mean best results.

Dataset We used seven bioinformatics data sets [32]: PTC, MUTAG, NCI1, NCI109, PROTEINS, ENZYMES, DD; and four social networks data sets: IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI-5K.

Hyperparamter Tuning For a d-dimension feature vector we chose $\sqrt{d}$ features to build a tree for efficient classification. We empirically chose the range of numbers of trees to be from set $\{50, 100, 500\}$. Gini impurity is used as a metric to build the tree and no constrained is imposed on the size of leaf nodes. Samples were bootstrapped and the number of samples to make a split in tree is used 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, giving us total of 14 graphs. While evaluating HOSD on larger datasets (Proteins, IMDb, and Reddit), we used a similar family up to four nodes, giving us a total of six graphs. This constraint was not necessary on HOSD-lite due to its scalability. We ran our experiments on this family and a depth of only two layers, showcasing HOSD and HOSD-lite on all verbosity levels discussed.

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	OMR	OMR	OMR	OMR	OMR	OMR	89.89%	89.46%	89.35%
REDDIT-MULTI-5K	OMR	OMR	OMR	OMR	OMR	OMR	52.41%	51.75%	51.13%

Table 2. Results of HOSD

Dataset	FGSD	NetLSD	NetSIMILIE	graph2vec	$HOSD_{G3}$	$HOSD\text{-}Lite_{M1}$
MUTAG	88.07%	85.28%	84.63%	69.34%	86.15%	87.87%
NCI1	79.60%	76.31%	73.96%	67.97%	80.31%	71.83%
NCI109	79.47%	76.12%	72.89%	66.51%	79.00%	70.52%
PTC_MR	58.91%	61.19%	59.09%	58.92%	61.06%	59.46%
PROTEINS	70.18%	74.63%	71.15%	70.55%	72.28%	74.45%
ENZYMES	33.95%	44.25%	42.10%	28.55%	49.87%	36.20%
DD	76.60%	77.27%	74.72%	75.68%	78.19%	77.10%
IMDB-BINARY	73.88%	73.79%	74.41%	59.96%	73.47%	70.85%
IMDB-MULTI	50.55%	50.57%	49.32%	46.43%	49.98%	47.94%
REDDIT-BINARY	81.60%	89.12%	89.53%	77.03%	OMR	89.89%
REDDIT-MULTI-5K	OMR	53.58%	52.78%	48.79%	OMR	52.41%

Table 3. Comparison with other methods

4.2 Classification Results

Table 2 shows the classification results on different data sets using different verbosities and types of sub-structures. After observing that most of the best results are obtained with a verbosity level of three on graphlets for HOSD, and similarly for motifs on HOSD-Lite. In table 3 we compare our method to four other graph descriptor paradigms: Graph2Vec, FGSD, NETSIMILIE, and NetLSD. Observe that HOSD outperforms more of them. Even for the datasets where HOSD falls short, **it is still within 1%** of the best. No importance of HOSD-lite here except that it is scalable.. Table 2 shows the classification results of HOSD and HOSD-lite, where G is for graphlet, M for motif and the subscript shows the verbosity. Comparison with other methods are shown in table 3. HOSD/HOSD-lite clearly outperform most of the state of the art methods. HOSD performs quite well in the bioinformatics data set and for social networks it is on par with others and it shows that HOSD works better if verbosity is set to 3 and HOSD-lite gives best result for verbosity 1. FGSD shows highest classification score MUTAG, NCI109 and IMDB-BINARY but its feature vectors are above $50,000$ which slows down the learning process. Alternate is to use HOSD/HOSD-lite's for these data sets, as results are quite close to FGSD and fall short less than 1%. HOSD also performs quite well on multi-class graph data sets like ENZYMES and IMDB-MULTI. Some lines to add that theory or underlying principles of HOSD are a good way to tackle the graph classfication. Just a simple line that concludes the paragraph and highlights the outstanding results of HOSD

5 Conclusion

This work presents the Higher Order Structure Descriptor: an interpretable, permutation invariant graph descriptor constructor that captures the structural information of a graph at multiple scales. HOSD has many degrees of freedom: number of layers, counting scheme, verbosity, and most importantly, choice of family. 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 unlabelled graph datasets, and offers results competitive with recently introduced state-of-the-art graph descriptors. Our future work involves generalizing HOSD to work with labeled graphs, using data-driven approaches to choosing an appropriate family, and catering to the running time that stymies its applicability.

Appendix

Appendix A: NP-Hardness of Familial Graph Compression

In this section we prove that in the general case it is NP-Hard to determine whether or not a family of graphs, $\mathcal{F}$, can completely compress a given graph, G. We begin by formally defining the problem:

Problem 3 (Familial Graph Compression (FGC)). Given a graph $G = (V, E)$, and a family of sub-graphs $\mathcal{F} = \{F_1, F_2, \ldots\}$, determine whether G can be compressed into one node via compressions of instances of $\mathcal{F}$ found in G. When a sub-graph, $F_i \in \mathcal{F}$, is chosen, all instances of F_i in G must be compressed. Instances of F_i that share node(s) will be compressed together. Sub-graphs, $F_i \in \mathcal{F}$, may be compressed infinitely many times, and in any order.

We will show that FGC is NP-Hard by providing a polynomial time reduction from the canonical NP-Hard problem, 3-SAT, which is 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 4 (3-Satisfiability (3-SAT)). Given a boolean expression in conjuctive normal form, where each clause has exactly three literals, determine whether there exists an assignment of variables such that the expression is satisfiable. The variables are from the set $X = \{x_1, x_2, \ldots\}$, and the clauses are from the set $C = \{C_1, C_2, \ldots\}$ where $x_{j_1}, x_{j_2}, x_{j_3}$ are the variables in C_j.

We reduce $3 - SAT$ to FGC by constructing G and $\mathcal{F}$ such that complete compression of G implies the given $3 - SAT$ instance is satisfiable.

Let h and g be functions such that $h : X \rightarrow \{1, 2, 3, \ldots\}$, and $g : C \rightarrow \{1, 2, 3, \ldots\}$:

$$h(x_i) = 4i + 1 \tag{2}$$

$$g(C_j) = j + 4|X| + 3 \tag{3}$$

Each variable, x_i, is assigned a graph with two cycles that share exactly one edge, as in Figure 5(b), where $k_1 = h(x_i)$ and $k_2 = h(x_i)+2$. Each clause, C_j, is assigned a graph as depicted in Figure 5(c); a tailed cycle on $g(C_j)$ nodes, which is connected to the graphs assigned to $x_{j_k}, \forall k \in \{1, 2, 3\}$. Finally, we construct G such that it contains all clause graphs, where the tailed node (depicted in red in Figure 5(c)) is shared amongst all clause graphs.

We will now construct our family such that G can only be completely compressed if the given 3-SAT instance is satisfiable.

For a variable, x_i, assume w.l.o.g. that compressing the cycle on $h(x_i)$ nodes assigns a value of True to x_i. Similarly, compressing the $(h(x_i) + 2)$-cycle assigns a value of False. To allow these compressions, cycles on $h(x_i)$ and $h(x_i) + 2$ nodes are added to $\mathcal{F}$, for all $x_i \in X$. A clause must be allowed to compress if a satisfiable assignment has been made for the variables within. For each clause, there are seven such assignments, so we include all seven in our family. To illustrate, Figure 5(d) depicts an example of some members of $\mathcal{F}$ corresponding to a clause.

Observe that G can only be completely compressed if all clause graphs can be compressed, analogous to how the boolean expression in 3-SAT is only satisfied if all clauses are satisfied. Conversely, if the 3-SAT instance is not satisfiable, the resultant G cannot be compressed using $\mathcal{F}$.

Clearly, the size of the graph G, family $\mathcal{F}$, and the size of the members of the family $F_i \in \mathcal{F}$ are polynomial in terms of $|X|$ and $|C|$. Thus, 3-SAT can be reduced to FGC in polynomial time, concluding that FGC is NP-Hard.

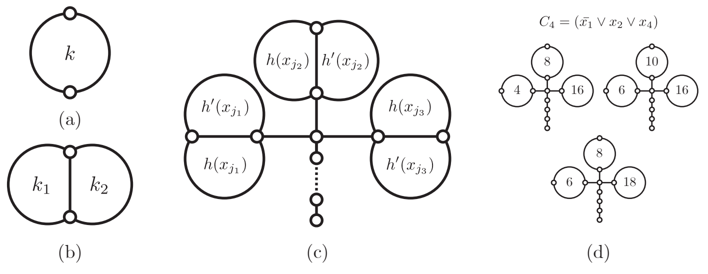

Fig. 5. (a) The figure represents a cycle on k nodes. The two explicit vertices are on opposite ends of the cycle. **(b)** The figure represents two cycles sharing exactly one edge on k_1 and k_2 nodes. The explicit nodes are the only two nodes on which both cycles are incident. **(c)** The subgraph in G corresponding to clause C_j, where $h'(x_i) = h(x_i)+2$. The dotted edge represents the remaining paths in the j-path assigned to C_j. **(d)** An example of the members of $\mathcal{F}$ corresponding to the clause written above.

References

1. Adelson, E.H., Anderson, C.H., Bergen, J.R., Burt, P.J., Ogden, J.M.: Pyramid methods in image processing. RCA engineer **29**(6), 33–41 (1984)
2. Aggarwal, C.C.: An introduction to social network data analytics. In: Social network data analytics, pp. 1–15. Springer (2011)
3. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Computing Surveys (CSUR) **40**(1), 1 (2008)
4. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1993–2001 (2016)
5. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Computer vision and image understanding **110**(3), 346–359 (2008)
6. Benkö, G., Flamm, C., Stadler, P.F.: A graph-based toy model of chemistry. Journal of Chemical Information and Computer Sciences **43**(4), 1085–1093 (2003)
7. Berlingerio, M., Koutra, D., Eliassi-Rad, T., Faloutsos, C.: Network similarity via multiple social theories. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. pp. 1439–1440. ACM (2013)
8. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Data Mining, Fifth IEEE International Conference on. pp. 8–pp. IEEE (2005)
9. Borgwardt, K.M., Petri, T., Vishwanathan, S., Kriegel, H.P.: An efficient sampling scheme for comparison of large graphs. Mining and Learning with Graphs, MLG pp. 1–3 (2007)
10. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
11. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering **30**(9), 1616–1637 (2018)
12. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub) graph isomorphism algorithm for matching large graphs. IEEE transactions on pattern analysis and machine intelligence **26**(10), 1367–1372 (2004)
13. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp. 3844–3852 (2016)
14. Dutta, A., Sahbi, H.: Stochastic graphlet embedding. IEEE Transactions on Neural Networks and Learning Systems pp. 1–14 (2018). https://doi.org/10.1109/TNNLS.2018.2884700
15. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in neural information processing systems. pp. 2224–2232 (2015)
16. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P., Kundu, M.: The journey of graph kernels through two decades. Computer Science Review **27**, 88–111 (2018)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR **abs/1609.02907** (2016)
18. Kondor, R., Pan, H.: The multiscale laplacian graph kernel. In: Advances in Neural Information Processing Systems. pp. 2990–2998 (2016)
19. Kozen, D.C.: Depth-first and breadth-first search. In: The Design and Analysis of Algorithms, pp. 19–24. Springer (1992)
20. Lindeberg, T.: Scale-space theory: A basic tool for analyzing structures at different scales. Journal of applied statistics **21**(1-2), 225–270 (1994)
21. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
22. Medeiros, C.B., Pires, F.: Databases for gis. ACM Sigmod Record **23**(1), 107–115 (1994)

23. Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., Saminathan, S.: subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. arXiv preprint arXiv:1606.08928 (2016)
24. Neuhaus, M., Riesen, K., Bunke, H.: 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). pp. 163–172. Springer (2006)
25. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International conference on machine learning. pp. 2014–2023 (2016)
26. Ramon, J., Gärtner, T.: Expressivity versus efficiency of graph kernels. In: First international workshop on mining graphs, trees and sequences. pp. 65–74. Citeseer (2003)
27. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.v., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(Sep), 2539–2561 (2011)
28. Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: Artificial Intelligence and Statistics. pp. 488–495 (2009)
29. Tsitsulin, A., Mottin, D., Karras, P., Bronstein, A.M., Müller, E.: Netlsd: Hearing the shape of a graph. In: KDD (2018)
30. Verma, S., Zhang, Z.L.: Hunt for the unique, stable, sparse and fast feature learning on graphs. In: Advances in Neural Information Processing Systems. pp. 88–98 (2017)
31. Wiener, H.: Structural determination of paraffin boiling points. Journal of the American Chemical Society **69**(1), 17–20 (1947)
32. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374. ACM (2015)