Strengthening structural baselines for graph classification using Local Topological Profile

Jakub Adamczyk[0000−0003−4336−4288] and Wojciech Czech[0000−0002−1903−8098]

AGH University of Science and Technology, Kraków, Poland
{jadamczy,czech}@agh.edu.pl

Abstract. We present the analysis of the topological graph descriptor Local Degree Profile (LDP), which forms a widely used structural baseline for graph classification. Our study focuses on model evaluation in the context of the recently developed fair evaluation framework, which defines rigorous routines for model selection and evaluation for graph classification, ensuring reproducibility and comparability of the results. Based on the obtained insights, we propose a new baseline algorithm called Local Topological Profile (LTP), which extends LDP by using additional centrality measures and local vertex descriptors. The new approach provides the results outperforming or very close to the latest GNNs for all datasets used. Specifically, state-of-the-art results were obtained for 4 out of 9 benchmark datasets. We also consider computational aspects of LDP-based feature extraction and model construction to propose practical improvements affecting execution speed and scalability. This allows for handling modern, large datasets and extends the portfolio of benchmarks used in graph representation learning. As the outcome of our work, we obtained LTP as a simple to understand, fast and scalable, still robust baseline, capable of outcompeting modern graph classification models such as Graph Isomorphism Network (GIN). We provide open-source implementation at GitHub.

Keywords: Graph representation learning · Graph classification · Fair evaluation · Graph descriptors · Baseline models.

1 Introduction

Graph classification is an essential variant of supervised learning problems, gaining popularity in many scientific fields due to the growing volume of structured datasets, which encode pairwise relations between modeled objects of different types. The applications of graph classification algorithms range from cheminformatics [11], where high-level properties of molecules such as toxicity or mutagenicity are predicted, to sociometry [29], biology [32] and technology [16], tackling different classes of complex networks, whose non-trivial dynamics can be explained by learning structural patterns.

Graph classification poses the inherent problem of measuring the dissimilarity between objects which do not lie in metric space but have combinatorial nature.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

This challenge is typically addressed by extracting isomorphism-invariant representations in the form of feature vectors [33] (also called graph embeddings, descriptors, fingerprints) or by constructing explicit pairwise similarity measures known as graph kernels [13]. More recently, the graph embedding problem was successfully reformulated within the framework of deep convolutional neural networks. Adopting the concept of convolution to vertex neighborhoods by introducing hierarchical iterative operators on multidimensional states of vertices allowed for building task-specific, low-dimensional representations for vertices, edges, and, after global pooling, the whole graph [27].

Baselines are the crucial elements of the fair comparison frameworks used in machine learning. As deep learning methods become increasingly powerful, the baseline algorithms used for their evaluation should also provide competitive results, forming good reference points for analyzing algorithms' performance. The recent development of Graph Neural Networks (GNNs), which automatically extract task-relevant features via deep learning, increased the number of attempts to solve various graph classification tasks [31,34]. Nevertheless, fair evaluation practices were frequently neglected in reported studies, and only recently the need for more rigorous model evaluation was highlighted [4]. This increased the demand for more powerful yet simple and fast baseline methods.

Motivated by recent findings regarding the discriminative power of Local Degree Profile (LDP) [2], which, together with SVM as the classification model, were proven to be competitive with the newest GNN models, we study the robustness and scalability of the new Local Topological Descriptor (LTP) built using histograms of specific descriptors representing vertex and edge structural features. We also use Random Forest classifier instead of SVM to reduce the sensitivity of baseline to hyperparameter tuning [19]. All experiments are performed in the regime of fair evaluation framework [4] to ensure replicability and to correct inaccuracies present in some earlier works (such as reporting accuracy on validation set). We also propose performance improvements in the implementation of LDP and LTP, resulting in better scalability and enabling the computation on large and dense social network benchmarks.

The key contributions of this work are the thorough analysis of the graph classification baseline composed of LDP and SVM, reporting limitations of this approach, the proposal of a new topological baseline utilizing Random Forest and experimental evaluation showing its robustness compared to the state-of-the-art. In addition, we present a modular software framework for LDP/LTP-based graph classification together with the associated open-source Python code shared on GitHub.

2 Related works

In the early works related to graph comparison, the concept of graph edit distance (GED) was introduced [1]. It was based on calculating the optimal sequence of elementary operations (adding/removing vertex/edge) required to transform one graph into another. Computational complexity prevented GED algorithms

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

from being widely used for larger graphs (> 1000 vertices). Nevertheless, multiple successful attempts at classifying attributed graphs were reported based on benchmark dataset [21]. The IAM graph database [21] formed the first consistent framework for comparing the efficiency of graph classification algorithms based on GED or graph embedding methods.

Graph embedding forms the comprehensive field of methods and applications aimed at the generation of multidimensional graph invariants/descriptors, which can be recognized as graph feature extraction or feature engineering [33]. Graph descriptors can be assigned to the vertex, the edge, or the graph itself. Representing a graph as a vector enables using a multitude of unsupervised and supervised machine learning algorithms suitable for tabular data. The most popular graph invariants come from the field of complex networks and spectral graph theory. They are represented by several graph centrality measures, *clustering coefficient*, *efficiency* and permutation invariants constructed from the eigenpairs of Laplace matrix [26]. Generic-purpose vertex and edge descriptors can be aggregated to form a high-dimensional graph representation such as B-matrix [3] or, after including vertex attributes, even more expressive relation order histograms [15]. Graph descriptors can be also extracted by mining frequent patterns/subgraphs, resulting in the topological fingerprint suitable for structural pattern recognition but also querying graph databases [14]. This approach was further extended by introducing domain-specific representations such as molecular fingerprints [22], which are widely used in the prediction of biochemical properties. As graph embedding algorithm can be adjusted to the domain, graph type (e.g. attributed vs. non-attributed), or even available computing resources, the topological descriptors still represent promising are for graph feature engineering and, as presented in [2], can compete with state-of-the-art graph representation learning techniques.

The concept of graph substructure mining was generalized in the form of graph kernels [13], which assess the structural similarity between two graphs by pairwise comparison of their subcomponents. Most typically, the concept of R-convolution [10] is applied as a generic purpose convolution framework for discrete structures. One of the most interesting representatives of this group are Weisfeiler-Lehman kernels designed for subtrees, edges, shortest paths, and whole graphs [23]. They utilize the concept of Weisfeiler-Lehman test of isomorphism, which was also used in the construction of Graph Isomorphism Networks (GIN). Another family of graph kernels uses the concept of optimal assignment [6] to reduce the number of pairwise sub-kernel computations required to obtain a similarity value. In this work, we focus on explicit graph embedding and graph representation learning, skipping graph kernels as a less feasible solution for scalable graph classification.

In case of graph embedding and graph kernels, the domain-specific knowledge can be incorporated by designing specific substructure descriptors with the help of experts. Such an approach can be treated as an example of feature engineering. The different method, providing automatic, task-specific graph feature extraction, is represented by Graph Representation Learning models exemplified by

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

Graph Neural Networks (GNNs). They form a modern and extensively studied framework for graph classification, with dozens of available models and specific taxonomy [18]. Graph Isomorphism Networks (GIN) [28] were designed to be as powerful as Weisfeiler-Lehman isomorphism test in discriminating graphs. They were reported to achieve state-of-the-art results on graph classification benchmarks; therefore, they will be used as the main reference point for evaluating our method. GraphSAGE [9] is a general inductive framework for different convolutional GNNs, providing a new neighborhood sampling method, which ensures fixed-size aggregation sets to limit computational overhead related to processing hubs, present, e.g., in social networks. The aggregation function for node states can be treated as a hyperparameter and tuned on the validation set. The newer DiffPool model [30] generates hierarchical graph representations using a differentiable graph pooling layer. It assigns nodes to clusters to achieve coarse-graining of input for the next layer, which reduces computation time. Edge-Conditioned Convolution (ECC) model [24] introduces convolutions over local graph neighborhoods using edge labels and custom coarsening procedure subsampling vertices on pooling layers to reduce graph size. The high classification accuracy was achieved by ECC on molecular datasets. Also, Deep Graph Convolutional Neural Network (DGCNN) model [31] proposes custom localized graph convolution similar to spectral filters and related to Weisfeiler-Lehman subtree kernel. Additionally, the new SortPooling layer is introduced, enabling standard neural network training on graphs. All models mentioned in this paragraph will be used in the evaluation of the new LTP baseline.

3 Methods

Graph classification tasks can be organized into well-defined pipeline. First, the graph is typically represented as a sparse adjacency matrix. Optionally, vertex and edge feature matrices can be used, if they are available. The matrices provide the input to the feature extraction algorithm, which outputs a feature vector representing a graph embedding in a metric space. Next, a tabular classification algorithm is used. For explicit feature extraction methods, such as LDP or graph kernels, the graph invariants are calculated by an algorithm distinct from the classifier. This allows using arbitrary algorithms for both parts. For graph representation learning methods, such as GNNs, those representations are typically learned end-to-end using a differentiable framework and gradient-based optimization, with multilayer perceptron (MLP) as a classifier. This potentially increases flexibility and makes embeddings more task-related, but requires vastly more data and computational resources.

Local Degree Profile (LDP) [2] proposes a feature extraction based on vertex degree statistics, which are calculated for each node in the graph and then aggregated into the embedding vector. Following conventions from [2], we denote the graph as $G(V, E)$, where V is the set of vertices (or nodes) v and E is the set of edges e. Degrees of neighboring nodes form a multiset $DN(v) = \{\text{degree}(v)|(u, v) \in E\}$. For each node, we then calculate the following

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

statistics: degree(v), min(DN), max(DN), mean(DN), std(DN). This way, for each node, we obtain the summary statistics of itself and its 1-hop neighborhood. They are then aggregated for the whole graph by calculating a histogram or empirical distribution function (EDF) over each feature. They are concatenated for all features, forming a final graph embedding. The number of bins used for aggregation and the choice between histogram and EDF are hyperparameters. There are also additional hyperparameters reflecting the method of preprocessing the features before the aggregation. Normalization can be applied: separately per graph, dividing the degrees by the highest value (this results in representing the feature value relative to the rest of the graph), or for the whole dataset, dividing by the highest degree in the dataset. Also, based on the observation that node degrees follow a power law for social networks, one can use log scale for aggregating features.

Any additional node- or edge-based structural descriptors can be included in the LDP in the form of histograms. The authors experiment with multiple ones: neighbors degree sum, lengths of shortest paths, closeness centrality, Fiedler vector and Ricci curvature. They remark that only shortest paths gave visible advantage, but could only be calculated in reasonable time on bioinformatics datasets, which have small molecular graphs. However, the gains using the shortest paths are consistent, about 2%, indicating that incorporating edge-based information can be beneficial. It should be noted that additional descriptors rapidly increase the dimensionality of the resulting embedding, which may result in degraded performance due to the curse of dimensionality, so only a limited number of well-chosen descriptors should be used.

Over the years, a vast number of node and edge descriptors were proposed. Among them, there are three commonly used groups, describing very different structural properties of the graph: centrality measures, link prediction indexes, and sparsification scores. Centrality indicates the importance or the influence of the node in the graph. Different measures focus on, e.g., how much information flows though a given node, or how many walks go through that node. They can also be defined for edges, indicating importance of connections in the graph. Link prediction indexes describe edges and are used to suggest the new edges to be added to the graph. They analyze the neighborhoods of the nodes, assigning high scores to the potential edges between nodes that share a large portion of their neighbors, or which have neighborhoods leading to shorter paths between them. Graph sparsification algorithms aim to eliminate edges, which are the least important for keeping the overall structure of the graph, especially in relation to hub nodes and communities. They aim to locally incorporate more global information about graph topology, assigning higher scores to more important edges.

During preliminary experiments, we surveyed descriptors representing each of those groups and available in the Networkit [25] library. While almost all gave promising results, we selected one from each group: edge betweenness centrality, Jaccard Index and Local Degree Score. The selection was based on intuitions that those particular descriptors will bring more edge-focused or more global

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

information than only node degree statistics, enhancing the discriminative power of the baseline.

Edge betweenness centrality (EBC) [7] is a centrality measure based on shortest paths, which measures how much influence the edge has over the flow of information in the network. It is defined as the fraction of the shortest paths in the graph going through the edge $e = (u, v)$:

$$EBC(e) = \sum_{s,t;(s,t)\neq(u,v)} \frac{\sigma_{st}(e)}{\sigma_{st}}, \tag{1}$$

where σ_{st} is the total number of shortest paths between nodes s and t, and $\sigma_{st}(e)$ is the number of those paths that go through e. This can be computed using Floyd-Warshall algorithm, which will give infinity values for disconnected graphs; we simply omit them in our implementation. We selected this descriptor, since it is based on shortest paths, which gave the good results in [2], and also takes into consideration the cyclic structure of the graph, e.g., it distinguishes molecules with linear scaffolds vs those with more ring-like topology.

Jaccard Index (JI) [12] is a normalized overlap between node neighborhoods $N(u)$ and $N(v)$:

$$JI(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{2}$$

We calculate it for the existing edges $e = (u, v)$ in the graph, obtaining a descriptor of a 3-hop subgraph. This feature should better discriminate between graph with visible community substructures and those without such node clusters.

Local Degree Score (LDS) [17] was proposed to detect edges between hubs, i.e. nodes with locally high degree, and keep only those edges after graph sparsification. For each node v, the rank of its neighbor u, $\mathrm{rank}(v, u)$ is the number of neighbors of v with degree lower than u. Note that this is asymmetrical, i.e. $\mathrm{rank}(u, v) \neq \mathrm{rank}(v, u)$. For each edge, the Local Degree Score is defined as:

$$LDS(e) = \max\left(1 - \frac{\ln \mathrm{rank}(v, u)}{\ln \mathrm{degree}(v)}, 1 - \frac{\ln \mathrm{rank}(u, v)}{\ln \mathrm{degree}(u)}\right), \tag{3}$$

which is simply taking the higher value from perspective of u or v, since either of them can be the hub node, giving a high LDS value. We selected this feature, since it can indicate the dispersion of nodes in the graph. If there are few edges with high LDS, it indicates that there are a few well separated clusters in the graph, centered around hub nodes, e.g., communities or well-connected functional groups in chemistry.

We propose to use the LDP descriptors together with the additional features described above, creating the **Local Topological Profile**. This method incorporates additional graph topology information in a local fashion, enhancing LDP with more discriminative power. Of course, this increases the computational cost, which we discuss below.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

LDP authors remark that shortest path lengths are not used for social network datasets due to unreasonably long computing time. Their implementation, however, uses NetworkX [8] for computing graph descriptors, which is a Python library, performing sequential computations. Instead, we propose to use Networkit [25], a parallelized C++ library. This way, we are able to utilize modern CPUs with multiple cores and compute descriptors in parallel. We do not perform a timing comparison, as we also could not finish computation with NetworkX in any reasonable time. The experiments on subsets of datasets indicate that Networkit is at least a one or two degrees of magnitude faster even on much smaller, molecular graphs. For this reason, our whole implementation of descriptors computation is based on Networkit.

Classification algorithm applied to feature vectors has a direct influence on both accuracy and scalability. LDP used Support Vector Machine (SVM) with a Gaussian kernel, which is a powerful classifier traditionally used with graph kernels, since they work well with small datasets. However, they are not scalable, since kernel calculation alone takes $O(n^2)$ for n graphs in the dataset. Moreover, they are sensitive to hyperparameter choice [19], hence requiring extensive tuning to obtain good results. In addition, they are typically trained with the Sequential Minimal Optimization (SMO) algorithm, which is inherently sequential, not utilizing modern CPUs with multiple cores. Linear SVMs, while faster to compute, typically give worse results, which the authors of LDP also observe.

We propose to change SVM to a Random Forest (RF) classifier. It is bagging ensemble of decision trees, which means that each tree can be trained independently in parallel, increasing scalability. Decision tree induction is also very fast, relying on a greedy top-down algorithm. They typically give good results with default hyperparameters, requiring only a sufficiently high number of trees [20]. In preliminary study, we did not observe any significant effect of hyperparameter tuning, even with large hyperparameter grids, hence we skipped this step.

More importantly, in contrast to LDP [2], we use a fair evaluation protocol with test sets. We use the fair comparison procedure from [4], adapted in the following way. The datasets and their statistics are summarized in Table 1. We use the same test splits, and apply 10-fold CV for testing. However, since our baseline is fast to compute, we can afford to perform inner 5-fold CV for validation and hyperparameter tuning, instead of holdout. Following [4], we report mean and standard deviation of accuracy on test sets.

Table 1. Statistics of datasets used, following [4].

Dataset	# Graphs	Avg. # Nodes	Avg. # Edges	# Classes
DD	1178	284.32	715.66	2
NCI1	4110	29.87	32.30	2
PROTEINS	1113	39.06	72.82	2
ENZYMES	600	32.63	64.14	6
IMDB-B	1000	19.77	96.53	2
IMDB-M	1500	13.00	65.94	3
REDDIT-B	2000	429.63	497.75	2
REDDIT-5K	4999	508.82	594.87	5
COLLAB	5000	74.49	2457.78	3

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

We tune the following hyperparameters for LDP (the same as the authors of [2]):

- number of bins: [30, 50, 70, 100]
- aggregation: [histogram, EDF]
- normalization: [none, graph, dataset]
- log scale: [false, true]

We tune the following hyperparameters for SVM (the same as authors of [2]):

- C (regularization): [10^{-3}, 10^{-2}, ..., 10^{2}, 10^{3}]
- γ (Gaussian kernel bandwidth): [10^{-2}, 10^{-1}, ..., 10^{1}, 10^{2}]

For RF, we do not perform tuning, instead setting the following parameters (based on Scikit-learn defaults) for dataset with n samples and d features: 500 trees, minimizing *Gini impurity*, using $\sqrt{d}$ features, sampling n samples with replacement.

We use PyTorch Geometric [5] for data loading and computing node degree features, Networkit [25] for computing EBC, JI and LDS descriptors, and Scikit-learn to implement SVM and RF. We perform all experiments using 12th Gen Intel Core i7-12700KF 3.61 GHz processor with 32 GB of RAM. Feature extraction processes graphs sequentially, while feature calculation is done in parallel, using all available cores. We use all available cores for RF (`n_jobs=-1`) and for grid search. We performed experiments to answer the following questions:

1. Can we improve training speed and prediction accuracy, using RF instead of kernel SVM? If so, by how much?
2. Is tuning all LDP hyperparameters necessary? Can we eliminate some hyperparameters, or set reasonable defaults, in order to decrease tuning time?
3. Do additional descriptors increase prediction accuracy? Can we use all 3 additional descriptors to get the best average improvement?
4. What is the difference in training speed between the original LDP (using SVM and with hyperparameter tuning) and our proposed LTP (using RF and without tuning)?
5. How does LTP compare against baselines from [4] and GNNs?

4 Results and discussion

The first experiment concerned comparison of LDP prediction accuracy when using RF (without tuning) instead of SVM (with tuning). We include both linear and kernel SVM results. We used a reasonable default values based on LDP paper [2]: 50 bins, histogram aggregation, normalization per graph, and linear scale. As shown in Table 2, RF always gave better results than both linear and kernel SVM. Interestingly, in some cases linear SVM outperformed kernel SVM, contradicting findings in [2]. The average improvement of RF over SVMs across all datasets is 3.4%, but can be as high as 7.3% on ENZYMES or 5.9% on DD. Additionally, the timings presented in Figure 1 indicate that RF is about an order of magnitude faster than SVM, being the result of both a more parallelizable

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

Table 2. Classification accuracy on testing sets using LDP features and three analyzed models. The best result for each dataset is marked in bold.

Dataset	Linear SVM	Kernel SVM	RF
DD	68.2 ± 4.3	68.9 ± 4.0	**74.9 ± 3.4**
NCI1	65.8 ± 2.7	71.5 ± 2.8	**73.8 ± 2.0**
PROTEINS	66.6 ± 3.2	66.0 ± 3.3	**71.1 ± 3.1**
ENZYMES	25.7 ± 6.0	29.5 ± 5.2	**36.8 ± 5.8**
IMDB-B	60.2 ± 4.3	64.3 ± 3.6	**65.9 ± 2.2**
IMDB-M	39.6 ± 3.4	35.3 ± 2.6	**43.9 ± 2.4**
REDDIT-B	78.0 ± 2.7	88.1 ± 2.1	**89.6 ± 1.5**
REDDIT-5K	46.4 ± 2.0	52.3 ± 1.5	**52.8 ± 1.4**
COLLAB	68.0 ± 2.3	71.0 ± 2.1	**73.5 ± 2.2**

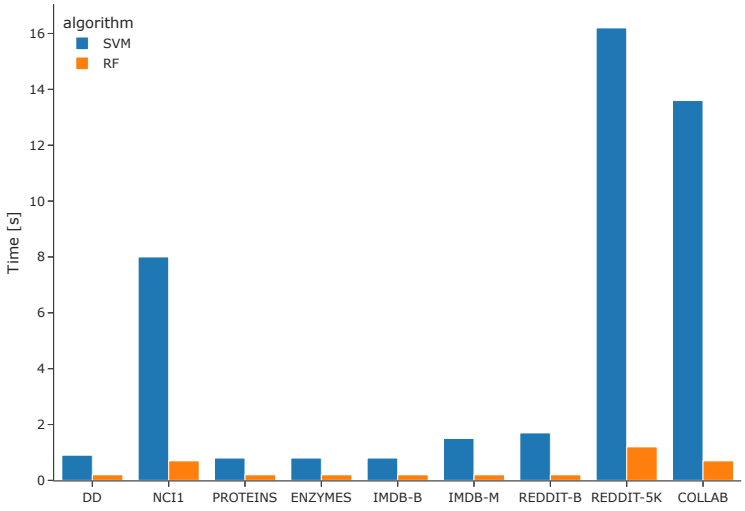

Fig. 1. Training time using LDP features: SVM vs RF clasifier.

algorithm and no need for hyperparameter tuning. Based on this finding, we only use RF in further experiments.

To verify the necessity of tuning LDP hyperparameters, we set the default values and vary a single hyperparameter at a time. We used 50 bins, histogram aggregation, normalization per graph, and linear scale. Due to space constraints, we do not include the whole results tables, but they are available on GitHub. For each hyperparameter, we calculate the number of times each value gave the best result. Additionally, for each value, we also calculate the absolute average difference between its result and the best result for a given hyperparameter on each dataset. The lower the absolute difference, the better, since it means that a given hyperparameter value, on average, gives the best results among all its possible values. results are presented in Table 3.

For the *number of bins*, we can clearly select 50 bins as the optimal value. While 30 bins gave the best results the same number of times, on average they performed worse compared to the optimal hyperparameter value. Similarly, for

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

normalization it is evident that we do not need to perform any kind of normalization, since using no normalization obtained the best results on majority of datasets and on average. This is somewhat contrary to the results obtained in LDP paper [2], but it is apparently an advantage of RF, since it considers each feature separately, while calculating tree splits. For *aggregation* method, the results are very close, both for number of wins and average difference compared to the best result. In this case, the choice does not matter that much, and we choose the simpler histogram method. The linear *scale* obtained much better results on average than the log scale, so the choice is obvious. Overall, this means that we can confidently recommend default values for all LDP hyperparameters, and tuning them is not particularly helpful. This dramatically decreases the computational cost, while having little effect on accuracy on average, which is a desirable tradeoff in a baseline method.

Table 3. Number of wins and absolute average difference between the result for a given hyperparameter value and the best result for any hyperparameter value. Higher number of wins is better, lower absolute average difference is better. For each hyperparameter, the value with the lowest absolute average difference has been marked in bold.

Hyperparameter	**Value**	**# Wins**	**Abs. avg. difference compared to best**
Number of bins	30	4	0.69%
	50	**4**	**0.22%**
	70	1	0.70%
	100	0	0.63%
Normalization	**None**	**5**	**0.23%**
	Graph	2	2.03%
	Dataset	2	0.39%
Aggregation	Histogram	5	0.71%
	EDF	**4**	**0.69%**
Scale	**Linear**	**4**	**0.35%**
	Log	5	0.64%

To assess whether additional descriptors increase accuracy of this method, we performed another set of experiments. We start with basic LDP, and add one additional descriptor at a time: lengths of shortest paths (SP), edge betweenness centrality (EBC), Jaccard Index (JI) and Local Degree Score (LDS). Finally, we check our proposed Local Topological Profile (LTP) method, combining LDP with EBC, JI and LDS descriptors. In all experiments, we keep the same hyperparameters: 50 bins, histogram aggregation, no normalization, and linear scale. As shown in Table 4, in every case the additional descriptors achieved the best result, while LTP was the best on 6 out of 9 datasets. On PROTEINS and IMDB-M it was the second best, being worse than the best by just 0.1% on the latter. It was also the third best on NCI1. Therefore, we can conclude that adding selected descriptors definitely increases the discriminatory power of this method. LTP is a robust method, performing the best on average, and using it eliminates the need to tune descriptor selection.

For performance analysis, we compare the original method from [2], i.e., LDP + SVM + hyperparameter tuning, with the proposed method, i.e. LTP + RF,

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

without any tuning. For the former, we tune feature extraction hyperparameter and SVM hyperparameters separately, since grid search on combined parameters grids would result in approximately 20 times larger number of models to be trained for a single test fold, which is infeasible. We measure the time for the whole experimental procedure, i.e. feature extraction and classifier training and tuning for all 10 test folds. This way, we take all characteristics of both approaches into consideration: LDP taking more time due to tuning, and LTP due to extracting more features. As shown in Figure 2, our proposed LTP approach is vastly superior to LDP in terms of speed, being 1–3 orders of magnitude faster on all datasets. It should be noted that our LDP implementation is nevertheless much faster than the original one, since we use PyTorch Geometric to compute LDP features in parallel with optimized C++ subroutines. The original Python-based, sequential implementation in NetworkX would be additionally 1–2 orders of magnitude slower, based on preliminary experiments. Our method was also very fast on datasets with large number of large graphs (REDDIT datasets and COLLAB), which indicates good scalability. This is especially important considering that graph datasets are getting larger and baselines also have to scale well.

Lastly, we compare accuracy of LTP to GNNs from [4], based on the same fair evaluation framework (compatible settings for model selection and model evaluation). For social networks, we compare against stronger models, using node degree. The outcome is summarized in Table 5. For easier comparison, in Table 6 we also present the average rank of the model across all datasets, i.e. on average which place, from 1st to 8th, it took. Our LTP approach achieves state-of-the-art results on IMDB-B, IMDB-M, REDDIT-B and COLLAB, achieving as much as 3.8% higher accuracy on COLLAB than the previous best method, GIN. Note that our method makes use of graph topology exclusively, ignoring node and edge features. This explains why on bioinformatics datasets we did not get as good results. In fact, on DD, PROTEINS and ENZYMES the best result is achieved by exclusively feature-based baseline from [4], which does not use graph topology at all. On average, LTP obtained the best rank among all models, beating even a theoretically very powerful GIN architecture. Additionally, all GNNs require GPUs and many hours of computation, while our method gives results in mere seconds.

5 Conclusions

We presented the new structural baseline for graph classification called Local Topological Profile (LTP). The research questions addressing its efficiency and scalability in comparison to related LDP method and competitive GNN methods were studied in the experimental section, where we conclude that using the Random Forest classifier instead of SVM improved the accuracy and the speed of computation by a large margin and this observation applies to all datasets used. We note that tuning of feature extraction hyperparameters is not necessary therefore, we can use default values for all datasets, decreasing tuning time sig-

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

Table 4. Classification accuracy of LDP with additional descriptors and LTP. The best result for each dataset is marked in bold.

Dataset	LDP	LDP + SP	LDP + EBC	LDP + JI	LDP + LDS	LTP
DD	76.0 ± 3.0	76.3 ± 2.8	77.0 ± 3.6	76.0 ± 3.4	75.8 ± 2.6	**77.1 ± 3.7**
NCI1	77.2 ± 1.5	76.1 ± 1.6	76.8 ± 1.7	76.6 ± 1.4	**77.4 ± 1.6**	77.0 ± 1.9
PROTEINS	70.6 ± 1.7	71.9 ± 2.2	**73.0 ± 3.2**	72.6 ± 3.2	71.4 ± 3.0	72.7 ± 4.2
ENZYMES	37.4 ± 4.0	37.2 ± 5.4	40.2 ± 6.5	40.0 ± 6.6	38.7 ± 5.6	**42.5 ± 4.1**
IMDB-B	71.3 ± 3.3	72.2 ± 4.0	72.9 ± 4.6	73.0 ± 4.3	74.2 ± 4.2	**74.5 ± 4.3**
IMDB-M	49.0 ± 4.4	49.2 ± 4.1	49.2 ± 5.0	49.3 ± 4.5	**50.1 ± 4.8**	50.0 ± 4.6
REDDIT-B	89.6 ± 1.2	90.5 ± 2.1	90.1 ± 1.7	89.6 ± 1.3	91.1 ± 1.1	**91.1 ± 1.0**
REDDIT-5K	51.9 ± 1.6	51.9 ± 1.9	51.7 ± 1.8	52.7 ± 2.0	53.1 ± 1.9	**53.3 ± 1.5**
COLLAB	75.7 ± 2.0	76.5 ± 2.2	76.8 ± 1.9	76.8 ± 1.8	78.7 ± 2.4	**79.4 ± 2.5**

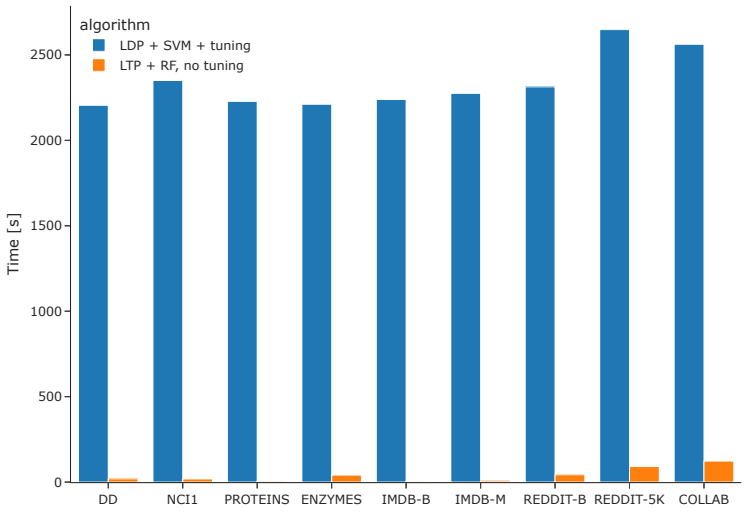

Fig. 2. Experiment time using LDP and LTP approaches.

nificantly. More importantly, we observe that introducing additional topological descriptors increases predictive accuracy in LTP method significantly. Using all three proposed descriptors (Edge Betweenness Centrality, Jaccard Index, Local Degree Score) gives very good prediction results across nine benchmark datasets, at the same time LTP is 2–3 orders of magnitude faster than the original LDP approach. Finally, we achieve state-of-the-art results on 4 out of 9 benchmark datasets, and in other cases get very strong accuracy, comparable to or even outcompeting modern GNNs, while using exclusively the graph topology. We share the software package with the research community, hoping that it can be useful in comparing results achieved by state-of-the-art graph classification models.

In our future work, we plan to extend the number of vertex/edge descriptors and enrich the expressive power of LTP towards more global features such as *eccentricity*. We also plan to merge LTP feature extraction and baselines from [4], to strengthen performance on more feature-focused bioinformatics datasets.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

Table 5. Comparison of accuracy with fair comparison results from [4]. Higher is better. Best result for each dataset has been marked in bold.

Dataset	Baseline [4]	DGCNN	DiffPool	ECC	GIN	GraphSAGE	LDP	LTP
DD	**78.4 ±4.5**	76.6 ±4.3	75.0 ±3.5	72.6 ±4.1	75.3 ±2.9	72.9 ±2.0	76.0 ±3.0	77.1 ±3.7
NCI1	69.8 ±2.2	76.4 ±1.7	76.9 ±1.9	76.2 ±1.4	**80.0 ±1.4**	76.0 ±1.8	77.2 ±1.5	77.0 ± 1.9
PROTEINS	**75.8 ±3.7**	72.9 ±3.5	73.7 ±3.5	72.3 ±3.4	73.3 ±4.0	73.0 ±4.5	70.6 ±1.7	72.7 ±4.2
ENZYMES	**65.2 ±6.4**	38.9 ±5.7	59.5 ±5.6	29.5 ±8.2	59.6 ±4.5	58.2 ±6.0	37.4 ±4.0	42.5 ±4.1
IMDB-B	70.8 ±5.0	69.2 ±3.0	68.4 ±3.3	67.7 ±2.8	71.2 ±3.9	68.8 ±4.5	71.3 ±3.3	**74.5 ±4.3**
IMDB-M	49.1 ±3.5	45.6 ±3.4	45.6 ±3.4	43.5 ±3.1	48.5 ±3.3	47.6 ±3.5	49.0 ±4.4	**50.0 ±4.6**
REDDIT-B	82.2 ±3.0	87.8 ±2.5	89.1 ±1.6	OOR	89.9 ±1.9	84.3 ±1.9	89.6 ±1.2	**91.1 ±1.0**
REDDIT-5K	52.2 ±1.5	49.2 ±1.2	53.8 ±1.4	OOR	**56.1 ±1.7**	50.0 ±1.3	51.9 ±1.6	53.3 ±1.5
COLLAB	70.2 ±1.5	71.2 ±1.9	68.9 ±2.0	OOR	75.6 ±2.3	73.9 ±1.7	75.7 ±2.0	**79.4 ±2.5**

Table 6. Comparison of average model ranks. The best result is marked in bold.

	Baseline [4]	DGCNN	DiffPool	ECC	GIN	GraphSAGE	LDP	LTP
Average rank	3.8	5.2	4.6	7.6	2.7	5.4	4	**2.6**

Acknowledgements The research presented in this paper was financed from the funds assigned by Polish Ministry of Science and Higher Education to AGH University of Science and Technology. We would like to thank Alexandra Elbakyan for her work and support for accessibility of science.

References

1. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern recognition letters **18**(8), 689–694 (1997)
2. Cai, C., Wang, Y.: A simple yet effective baseline for non-attributed graph classification. arXiv preprint arXiv:1811.03508 (2018)
3. Czech, W.: Invariants of distance k-graphs for graph embedding. Pattern Recognition Letters **33**(15), 1968–1979 (2012)
4. Errica, F., Podda, M., Bacciu, D., Micheli, A.: A fair comparison of graph neural networks for graph classification. arXiv preprint arXiv:1912.09893 (2019)
5. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
6. Fröhlich, H., Wegner, J.K., Sieker, F., Zell, A.: Optimal assignment kernels for attributed molecular graphs. In: Proceedings of the 22nd international conference on Machine learning. pp. 225–232 (2005)
7. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
8. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

9. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
10. Haussler, D., et al.: Convolution kernels on discrete structures. Tech. rep., Citeseer (1999)
11. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. Advances in neural information processing systems **33**, 22118–22133 (2020)
12. Jaccard, P.: The distribution of the flora in the alpine zone. 1. New phytologist **11**(2), 37–50 (1912)
13. Kriege, N.M., Johansson, F.D., Morris, C.: A survey on graph kernels. Applied Network Science **5**(1), 1–42 (2020)
14. Kuramochi, M., Karypis, G.: An efficient algorithm for discovering frequent subgraphs. IEEE transactions on Knowledge and Data Engineering **16**(9), 1038–1051 (2004)
15. Łazarz, R., Idzik, M.: Relation order histograms as a network embedding tool. In: Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part II 21. pp. 224–237. Springer (2021)
16. Li, Y., Gu, C., Dullien, T., Vinyals, O., Kohli, P.: Graph matching networks for learning the similarity of graph structured objects. In: International conference on machine learning. pp. 3835–3845. PMLR (2019)
17. Lindner, G., Staudt, C.L., Hamann, M., Meyerhenke, H., Wagner, D.: Structure-preserving sparsification of social networks. In: Proceedings of the 2015 IEEE/ACM International conference on advances in social networks analysis and mining 2015. pp. 448–454 (2015)
18. Liu, R., Cantürk, S., Wenkel, F., Sandfelder, D., Kreuzer, D., Little, A., McGuire, S., O'Bray, L., Perlmutter, M., Rieck, B., et al.: Taxonomy of benchmarks in graph representation learning. arXiv preprint arXiv:2206.07729 (2022)
19. Probst, P., Boulesteix, A.L., Bischl, B.: Tunability: Importance of hyperparameters of machine learning algorithms. The Journal of Machine Learning Research **20**(1), 1934–1965 (2019)
20. Probst, P., Wright, M.N., Boulesteix, A.L.: Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Reviews: data mining and knowledge discovery **9**(3), e1301 (2019)
21. Riesen, K., Bunke, H., et al.: Iam graph database repository for graph based pattern recognition and machine learning. In: SSPR/SPR. vol. 5342, pp. 287–297 (2008)
22. Rogers, D., Hahn, M.: Extended-connectivity fingerprints. Journal of chemical information and modeling **50**(5), 742–754 (2010)
23. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(9) (2011)
24. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3693–3702 (2017)
25. Staudt, C.L., Sazonovs, A., Meyerhenke, H.: Networkit: A tool suite for large-scale complex network analysis. Network Science **4**(4), 508–530 (2016)
26. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern vectors from algebraic graph theory. IEEE transactions on pattern analysis and machine intelligence **27**(7), 1112–1124 (2005)
27. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47

28. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
29. 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 (2015)
30. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems **31** (2018)
31. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
32. Zhang, X.M., Liang, L., Liu, L., Tang, M.J.: Graph neural networks and their current applications in bioinformatics. Frontiers in genetics **12**, 690049 (2021)
33. Zhang, Y.J., Yang, K.C., Radicchi, F.: Systematic comparison of graph embedding methods in practical tasks. Physical Review E **104**(4), 044315 (2021)
34. Zhou, Y., Zheng, H., Huang, X., Hao, S., Li, D., Zhao, J.: Graph neural networks: Taxonomy, advances, and trends. ACM Transactions on Intelligent Systems and Technology (TIST) **13**(1), 1–54 (2022)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_47