TUDataset: A collection of benchmark datasets for learning with graphs

Christopher Morris [1] **Nils M. Kriege** [2] **Franka Bause** [3] **Kristian Kersting** [4] **Petra Mutzel** [5] **Marion Neumann** [6]

Abstract

Recently, there has been an increasing interest in (supervised) learning with graph data, especially using graph neural networks. However, the development of meaningful benchmark datasets and standardized evaluation procedures is lagging. That is, most papers still evaluate their methods on small-scale datasets leading to high standard deviations and hard to interpret results, consequently hindering advancements in this area. To address this, we introduce the TUDATASET for graph classification and regression. The collection consists of over 120 datasets of varying sizes from a wide range of applications. We provide Python-based data loaders, kernel, and graph neural network baseline implementations, and evaluation tools. Here, we give an overview of the datasets, standardized evaluation procedures, and provide baseline experiments. All datasets are available at www.graphlearning.io. The experiments are fully reproducible from the code available at www.github.com/chrsmrrs/tudataset.

1. Introduction

Graph-structured data is ubiquitous across application domains ranging from chemo- and bioinformatics (Barabasi & Oltvai, 2004; Stokes et al., 2020) to image (Simonovsky & Komodakis, 2017) and social network analysis (D. & J., 2010). To develop successful machine learning models in these domains, we need techniques that can exploit the rich information inherent in the graph structure, as well as the feature information contained within nodes and edges. In recent years, numerous approaches have been proposed for machine learning with graphs—most notably, approaches based on *graph kernels* (Kriege et al., 2020) and *graph neural networks* (GNNs) (Scarselli et al., 2009; Gilmer et al., 2017). However, most papers, even recent ones, evaluate newly proposed architectures or methods on a fixed set of small-scale benchmark datasets leading to high standard deviations and hard to interpret results. Moreover, non-standardized experimental setups are used, which hinders the comparison of results from different publications.

Present work. Here, we give an overview of TUDATASET. The benchmark collection consists of over 120 datasets from a wide range of domains for supervised learning with graphs, i.e., classification and regression. All datasets are provided in a standard dataset format at www.graphlearning.io and are easily accessible from popular graph learning frameworks such as PYTORCH GEOMETRIC (Fey & Lenssen, 2019)[1] and DGL (Wang et al., 2019)[2]. To facilitate a standard comparison of kernel and neural approaches, we provide implementations of standard algorithms and easy-to-use evaluation procedures. Moreover, we report results on an experimental study comparing graph kernels and GNNs on a subset of the TUDATASET.

Related work. There exist two main approaches to supervised learning with graphs, graph kernels and graph neural networks (GNNs). Graph kernels have been studied extensively in the past 15 years, see (Kriege et al., 2020) for a thorough overview. Important approaches include random-walk and shortest paths based kernels (Gärtner et al., 2003; Sugiyama & Borgwardt, 2015; Borgwardt & Kriegel, 2005; Kriege et al., 2019), as well as the Weisfeiler-Lehman subtree kernel (Shervashidze et al., 2011; Morris et al., 2017). Further recent works focus on approaches based on assignments (Kriege et al., 2016; Nikolentzos et al., 2017), spectral properties (Kondor & Pan, 2016), graph decomposition (Nikolentzos et al., 2018), randomized binning (Heimann et al., 2019), and the extension of kernels based on the Weisfeiler-Leman algorithm (Togninalli et al., 2019; Rieck et al., 2019). For a theoretical investigation of graph kernels, see (Kriege et al., 2018b). Recently, graph neural networks (GNNs) (Gilmer et al., 2017; Scarselli et al., 2009) emerged

[1]CERC in Data Science for Real-Time Decision-Making, Polytechnique Montréal [2]Faculty of Computer Science, University of Vienna [3]Department of Computer Science, TU Dortmund University [4]Machine Learning Group, TU Darmstadt [5]Department of Computer Science, University of Bonn [6]Department of Computer Science and Engineering, Washington University in St. Louis. Correspondence to: Christopher Morris <christopher.morris@tu-dortmund.de>.

Graph Representation Learning and Beyond (GRL+), ICML 2020 Workshop.

[1]https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html

[2]https://docs.dgl.ai/en/0.4.x/api/python/data.html

as an alternative to graph kernels. Notable instances of this architecture include, e.g., (Duvenaud et al., 2015; Hamilton et al., 2017; Velickovic et al., 2018), and the spectral approaches proposed in, e.g., (Bruna et al., 2014; Defferrard et al., 2016; Kipf & Welling, 2017; Monti et al., 2017)—all of which descend from early work in (Kireev, 1995; Merkwirth & Lengauer, 2005; Sperduti & Starita, 1997; Scarselli et al., 2009). A survey of recent advancements in GNN techniques can be found, e.g., in (Chami et al., 2020; Wu et al., 2019; Zhou et al., 2018).

The papers (Fey & Lenssen, 2019; Chen et al., 2019b; Errica et al., 2019; Dwivedi et al., 2020) evaluate GNNs using a unified evaluation procedure, however, they only use small- or medium scale datasets. Recently, `ogb.stanford.edu` (Hu et al., 2020) launched, however, the provided datasets for graph classification focus on chemistry and bioinformatic applications, and the number is quite limited at this point. Moreover, the datasets proposed in (Ferber et al., 2019) focuses on instances from planning competitions. Recent efforts to implement graph kernels in a common framework such as the GRAKEL library (Siglidis et al., 2018) foster comparability, but do not solve the dataset related issues discussed above, and only focus on kernel approaches.

2. The TUDataset collection

The TUDATASET collection contains over 120 datasets provided at `www.graphlearning.io`. The datasets, baseline methods and experimental evaluation tools can be conveniently accessed from the Python interface, see Appendix A for further details. Dataset statistics and further documentation is available at our website.

2.1. Datasets

Our collection of datasets covers graphs from various domains, contributed by different authors. Therefore, they differ regarding the used graph model even within the same domain and the provided annotations, e.g., discrete or continuous node and edge attributes. Here, we give an overview of some representative domains and graph models.

Small molecules. A common class of graph datasets consists of small molecules with class labels representing, e.g., toxicity or biological activity determined in drug discovery projects. Here, a graph represents a molecule, i.e., nodes take the places of atoms and edges that of chemical bonds. Consequently, the labels encode atom and bond types, possibly with additional chemical attributes. The graph models differ, e.g., in whether hydrogen atoms are represented explicitly by nodes, and bonds in aromatic rings are annotated accordingly.

Our collection contains small datasets commonly used in the early graph kernel literature such as MUTAG (Debnath et al., 1991) and PTC (Helma et al., 2001), medium-sized datasets, e.g., NCI1 and NCI109 (Wale et al., 2008; Shervashidze et al., 2011), as well as several large datasets derived from the TOX21 challenge 2014 or PUBCHEM (Kim et al., 2018). This includes the eleven datasets from anticancer screen tests with different cancer cell lines used by Yan et al. (2008) to demonstrate the efficacy of classifiers based on significant graph patterns. These datasets, the largest of which contains more than 79k graphs, are typically not balanced and contain far more small molecules that are identified as inactive against cancer cells. Moreover, our collection also contains large-scale molecular regression tasks such as ALCHEMY (Chen et al., 2019a), QM9 (Ramakrishnan et al., 2014), and ZINC (Dwivedi et al., 2020; Jin et al., 2018). The first two contain 3D coordinates of the nodes, which should be taken into account in a rotation-invariant manner to benefit from the geometrical information.

Bioinformatics. The datasets DD, ENZYMES and PROTEINS represent macromolecules. Borgwardt et al. (2005) introduced a graph model for proteins, where nodes represent secondary structure elements and are annotated by their type, i.e., helix, sheet, or turn, as well as several physical and chemical information. An edge connects two nodes if they are neighbors along the amino acid sequence or one of three nearest neighbors in space. Using this approach, the dataset ENZYMES was derived from the BRENDA database (Schomburg et al., 2004). Here, the task is to assign enzymes to one of the 6 EC top-level classes, which reflect the catalyzed chemical reaction. Similarly, the dataset PROTEINS was derived from (Dobson & Doig, 2003), and the task is to predict whether a protein is an enzyme. The dataset DD used by Shervashidze et al. (2011) is based on the same data, but contains graphs, where nodes represent individual amino acids and edges their spatial proximity.

Computer vision. Graph-based methods are widely used in computer vision for various tasks using diverse graph models. Our collection provides several datasets originating from the IAM GRAPH DATABASE (Riesen & Bunke, 2008) such as LETTER and FINGERPRINT. Other datasets represent CUNEIFORM signs (Kriege et al., 2018a), 3D point clouds for robot grasping tasks (FIRSTMM_DB) and semantic image processing (MSRC) (Neumann et al., 2016).

Social networks. Yanardag & Vishwanathan (2015) introduced several graph classification datasets derived from social networks. In the REDDIT datasets, each graph represents a discussion thread, where nodes correspond to users, two of which are connected by an edge if one responded to a comment of the other. This graph model is used to derive several datasets, where the classification task is to distinguish either discussion-based and question/answer-based subreddits (REDDIT-BINARY) or predict the subreddit, where the thread was posted (REDDIT-MULTI-5K and REDDIT-MULTI-12K). COLLAB are datasets derived from scientific collaboration

networks. Each graph is the ego-network of a researcher, and the task is to predict their research field, i.e., high energy, condensed matter, or astrophysics. Similarly, the IMDB datasets consist of ego-networks derived from actor collaborations, and the task is to predict the genre, e.g., Action vs. Romance. Rozemberczki et al. (2020) used similar approaches to obtain more massive social network datasets. REDDIT_THREADS contains more than 200k graphs with the task to predict whether a thread is discussion-based. DEEZER_EGO_NETS and TWITCH_EGOS contain ego-networks derived from online services, and the task is to predict the gender and play behavior (single or multiple games) of the central user. GITHUB_STARGAZERS contains graphs representing the social networks of GitHub users divided into those who starred popular machine learning and web development repositories.

Recently, temporal graphs were considered by Oettershagen et al. (2019), where edges represent the contact or interaction between two individuals at a certain point in time. These graphs are of interest when studying dissemination processes such as the spreading of epidemics, rumours or fake news. We provide temporal graph classification datasets derived from TUMBLR (Rozenshtein et al., 2016), DBLP and FACEBOOK (Viswanath et al., 2009) as well as contacts between students at the MIT (Eagle & Pentland, 2006), in a HIGHSCHOOL and visitors at the INFECTIOUS exhibition (Isella et al., 2011).

Synthetic. Several graph datasets were generated to demonstrate the strengths or weaknesses of specific methods. The datasets SYNTHETICNEW and SYNTHIE were created by Feragen et al. (2013) (see Erratum) and Morris et al. (2016), respectively, to demonstrate the ability of kernels to operate on graphs with continuous attributes. Knyazev et al. (2019) introduced the datasets COLORS and TRIANGLES, where the task is to count the number of nodes with a given one-hot-encoded color and the number of triangles, respectively.

2.2. Baselines methods

To provide meaningful baselines, we provide implementations of common graph kernels as well as GNN architectures. We have implemented the *Weisfeiler-Lehman Subtree* (Shervashidze et al., 2011), *Shortest-path* (Borgwardt & Kriegel, 2005), *Graphlet* (Shervashidze et al., 2009) (using labeled subgraphs with three nodes), *Weisfeiler-Lehman Optimal Assignment* (Kriege et al., 2016) kernel in C++ and made them accessible through the Python interface of TUDATASET. Moreover, all GNN architectures provided by PYTORCH GEOMETRIC can be conveniently used as a baseline as well.

2.3. Evaluation methods

To ensure a fair and meaningful comparison between methods, we propose the following evaluation procedures for kernels and GNNs. First, for kernels, we propose the established C-SVM implementation LIBSVM (Chang & Lin, 2011) for kernels that compute a Gram matrix, and the linear C-SVM implementation LIBLINEAR (Fan et al., 2008) for kernels that can be computed based on sparse, explicit feature maps. We optimize GNNs end-to-end using ADAM (Kingma & Ba, 2015). To compute classification accuracies, we propose to use 10-fold cross-validation, where we select 10% of each training fold uniformly at random as validation set to optimize hyperparameters, e.g., the number of iterations, C parameter, number of layers, feature dimension. We repeat the above evaluation ten times and report standard deviations over all ten repetitions, and additionally across all one hundred runs (i.e., ten repetitions with ten folds each). For the large-scale molecule learning tasks, we either use random splits (80%/10%/10%) or the provided, fixed splits and report MAE (mean std. MAE, mean std. logMAE for multi-target regression, see (Klicpera et al., 2020)) over five runs.

3. Experimental evaluation

Our intent here is to provide baseline experiments and compare graph kernels and GNNs. We used the following datasets, graph kernels, and GNN baselines.

Datasets. We used the DEEZER_EGO_NETS, GITHUB_STARGAZERS, ENYMES, IMDB-BINARY, IMDB-MULTI, MCF-7, MOLT-4, NCI1, PROTEINS, REDDIT-BINARY, REDDIT_THREADS, TWITCH_EGOS, UACC257 graph classification datasets. Moreover, we used the ALCHEMY, QM9, ZINC (multi-target) regression datasets. See the website for dataset statistics. We opted for not using continuous node features of the small datasets (if available) and the 3D-coordinates of the ALCHEMY dataset to solely provide baseline results based on graph structure and discrete labels. In case of the QM9 dataset, we closely replicated the (continuous) node and edge features of Gilmer et al. (2017).

Graph kernels. As kernel baselines we used the *Weisfeiler-Lehman Subtree* (1-WL) (Shervashidze et al., 2011), *Shortest-path* (SP) (Borgwardt & Kriegel, 2005), *Graphlet* (GR) (Shervashidze et al., 2009), *Weisfeiler-Lehman Optimal Assigment* (WL-OA) (Kriege et al., 2016) included in the TUDATASET package. The C-parameter was selected from $\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$ from the validation set. For the larger datasets, we computed sparse feature vectors for each graph and used the linear C-SVM implementation of LIBLINEAR (Fan et al., 2008). The number of iterations of the 1-WL and WL-OA were selected from $\{0, \ldots, 5\}$.[3]

All kernel experiments were conducted on a workstation with an Intel Xeon E5-2690v4 with 2.60GHz and 384GB of RAM running Ubuntu 16.04.6 LTS using a single core.

[3] As already shown in (Shervashidze et al., 2011), choosing the number of iterations too large will lead to overfitting.

Table 1. Classification accuracies in percent and standard deviations on small-scale datasets.

	Method	Dataset					
		ENZYMES	IMDB-BINARY	IMDB-MULTI	NCI1	PROTEINS	REDDIT-BINARY
Kernel	1-WL	52.3 $\pm 0.7 \pm 6.7$	72.6 $\pm 0.6 \pm 4.1$	50.3 $\pm 0.8 \pm 3.8$	84.5 $\pm 0.3 \pm 1.7$	72.8 $\pm 0.8 \pm 3.9$	73.9 $\pm 0.6 \pm 3.2$
Kernel	WL-OA	58.4 $\pm 2.0 \pm 7.4$	73.2 $\pm 0.8 \pm 3.8$	49.4 $\pm 0.3 \pm 3.6$	85.4 $\pm 0.2 \pm 1.4$	73.2 $\pm 0.6 \pm 3.5$	88.4 $\pm 0.4 \pm 2.1$
Kernel	GR	30.1 $\pm 0.8 \pm 6.0$	59.0 $\pm 0.9 \pm 5.7$	39.6 $\pm 0.5 \pm 3.6$	66.2 $\pm 0.2 \pm 2.2$	71.4 $\pm 0.6 \pm 4.4$	60.0 $\pm 0.2 \pm 2.7$
Kernel	SP	40.5 $\pm 1.6 \pm 6.2$	58.7 $\pm 0.9 \pm 4.6$	39.5 $\pm 0.9 \pm 4.2$	74.5 $\pm 0.4 \pm 2.2$	75.9 $\pm 0.6 \pm 4.0$	84.6 $\pm 0.2 \pm 0.4$
GNN	GIN-ε	39.5 $\pm 1.4 \pm 7.6$	72.4 $\pm 0.6 \pm 4.0$	49.7 $\pm 0.2 \pm 3.9$	79.0 $\pm 0.5 \pm 1.9$	71.3 $\pm 0.4 \pm 4.0$	90.0 $\pm 0.6 \pm 1.7$
GNN	GIN-ε-JK	41.3 $\pm 1.8 \pm 6.7$	73.4 $\pm 0.6 \pm 4.4$	50.6 $\pm 0.5 \pm 4.2$	79.1 $\pm 0.2 \pm 1.5$	72.4 $\pm 0.6 \pm 1.7$	90.6 $\pm 0.2 \pm 2.2$

Table 2. Classification accuracies in percent and standard deviations on mid-scale datasets.

	Method	Dataset				
		MCF-7	MOLT-4	YEAST	GITHUB_STAR	REDDIT_THREADS
Kernel	1-WL	94.5 $< 0.1 \pm 0.4$	94.6 $\pm 0.1 \pm 0.3$	89.2 $\pm 0.1 \pm 0.4$	63.9 $\pm 0.1 \pm 1.3$	77.0 $< 0.1 \pm 0.3$
Kernel	GR	91.7 $< 0.1 \pm 0.5$	92.1 $< 0.1 \pm 0.4$	88.2 $< 0.1 \pm 0.4$	53.5 $< 0.1 \pm 1.3$	51.2 $< 0.1 \pm 0.4$
Kernel	SP	91.7 $< 0.1 \pm 0.4$	92.1 $< 0.1 \pm 0.4$	88.2 $< 0.1 \pm 0.4$	64.1 $\pm 0.1 \pm 1.4$	77.3 $< 0.1 \pm 0.3$
GNN	GINE-ε	92.0 $\pm 0.1 \pm 0.6$	92.3 $\pm 0.1 \pm 0.6$	88.3 $< 0.1 \pm 0.3$	67.1 $\pm 0.5 \pm 1.6$	77.2 $< 0.1 \pm 0.3$
GNN	GINE-ε-JK	91.8 $< 0.1 \pm 0.5$	92.2 $< 0.1 \pm 0.5$	88.2 $< 0.1 \pm 0.3$	67.2 $\pm 0.4 \pm 1.3$	77.3 $< 0.1 \pm 0.3$

Moreover, we used the GNU C++ Compiler 5.5.0 with the flag `-O2`.

GNNs. For comparison with kernel methods, we used GIN-ε (Xu et al., 2019) and GIN-ε-JK with jumping knowledge networks as neural baselines (Xu et al., 2018). For data with (continuous) edge features, we used a 2-layer MLP to map them to the same number of components as the node features and combined them using summation (GINE-ε and GINE-ε-JK). We used mean pooling to pool the learned node embeddings to a graph embedding and used a 2-layer MLP for the final classification, using a dropout layer with $p = 0.5$ after the first layer of the MLP. For the smaller datasets of Table 1, we optimized the number of hidden units from $\{32, 64, 128\}$, the number of layers from $\{1, 2, 3, 4, 5\}$ using the validation set. For the mid-scale datasets, due to computation time constraints, we set the number of hidden units to 64 and the number of layers to 3. Moreover, for both, we use a learning rate decay of 0.5 with a patience parameter of 5, a starting learning rate of 0.01 and a minimum of 10^{-6}, a maximum epoch number of 200. For both methods, we used the evaluation procedure described in Section 2.3 to optimize hyperparameters and compute accuracies. See the appendix for details on the hyperparameter and evaluation protocols used for the larger molecular regression tasks (ZINC, and ALCHEMY, QM9). All results are reproducible using the scripts provided at `https://github.com/chrsmrrs/tudatasets`.

Results and discussion. Tables 1 to 3 summarize the results. On the small-scale datasets, see Table 1, the WL-OA performs best overall, i.e., only on the IMDB-BINARY and the REDDIT-BINARY dataset it is beaten by the 1-WL or the neural baselines. However, it does not scale to large datasets, Table 2, as it relies on Gram matrix computation. Here, the 1-WL performs well on all datasets, excluding GITHUB_STARGAZERS, where the neural baselines perform best overall.[4] Our results show that despite the extensive research on GNNs in recent years, classical graph kernels in combination with SVMs are still highly competitive in graph classification tasks. On the large-scale molecular learning tasks, see Table 3, it becomes apparent that specialized architectures such as MPNN result in significant gains over the generic GINE-ε baseline.

Table 3. Mean MAE (mean std. MAE, logMAE) on large-scale (multi-target) molecular regression tasks.

Method	Dataset		
	ZINC	ALCHEMY	QM9
GINE-ε	0.084 ± 0.004	0.103 ± 0.001 -2.956 ± 0.029	0.081 ± 0.003 -3.400 ± 0.094
MPNN	—	—	0.034 ± 0.001 -4.156 ± 0.030

4. Conclusion

We gave an overview of the TUDATASET collection, and reported on the results of an experimental study comparing graph kernels and GNNs on a subset of the data. We believe that our dataset collection will spark further progress in graph representation learning, and that our unified evaluation procedures will improve the comparability of results. We are looking forward to adding more datasets and are excited about contributions from the community, researchers, and practitioners from other areas. Future work includes a more extensive comparision of kernel and neural approaches on large-scale molecular regression tasks with continuous node and edge features.

[4] For the neural baselines unlike the kernel baselines, we used one-hot degree features for datasets that do not provide node labels.

5. Acknowledgement

We thank everybody who provided datasets for the TU-DATASET collection.

This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", project A6 "Resource-efficient Graph Mining". Nils Kriege has been supported by the Vienna Science and Technology Fund (WWTF) through project VRG19-009.

References

Anderson, B. M., Hy, T., and Kondor, R. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems 32*, pp. 14510–14519, 2019.

Barabasi, A.-L. and Oltvai, Z. N. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

Borgwardt, K. M. and Kriegel, H.-P. Shortest-path kernels on graphs. In *IEEE International Conference on Data Mining*, pp. 74–81, 2005.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(Supplement 1):i47–i56, 2005.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and deep locally connected networks on graphs. In *International Conference on Learning Representation*, 2014.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., and Murphy, K. Machine learning on graphs: A model and comprehensive taxonomy. *CoRR*, abs/2005.03675, 2020.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

Chen, G., Chen, P., Hsieh, C., Lee, C., Liao, B., Liao, R., Liu, W., Qiu, J., Sun, Q., Tang, J., Zemel, R. S., and Zhang, S. Alchemy: A quantum chemistry dataset for benchmarking AI models. *CoRR*, abs/1906.09427, 2019a.

Chen, T., Bian, S., and Sun, Y. Are powerful graph neural nets necessary? A dissection on graph classification. *CoRR*, abs/1905.04579, 2019b.

D., E. and J., K. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34 (2):786–797, 1991.

Defferrard, M., X., B., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771 – 783, 2003.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pp. 2224–2232, 2015.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *CoRR*, abs/2003.00982, 2020.

Eagle, N. and Pentland, A. Reality Mining: Sensing complex social systems. *Personal Ubiquitous Computing*, 10(4): 255–268, 2006.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. *CoRR*, abs/1912.09893, 2019.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Feragen, A., Kasenburg, N., Petersen, J., Bruijne, M. D., and M., B. K. Scalable kernels for graphs with continuous attributes. In *Advances in Neural Information Processing Systems*, pp. 216–224. NIPS, 2013. Erratum available at http://image.diku.dk/aasa/papers/graphkernels_nips_erratum.pdf.

Ferber, P., Ma, T., Huo, S., Chen, J., and Katz, M. IPC: A benchmark data set for learning with graph-structured data. *CoRR*, abs/1905.06393, 2019.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. *CoRR*, abs/1903.02428, 2019.

Gärtner, T., Flach, P., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pp. 129–143. 2003.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1025–1035, 2017.

Heimann, M., Safavi, T., and Koutra, D. Distribution of node embeddings as multiresolution features for graphs. In *IEEE International Conference on Data Mining*, pp. 289–298, 2019.

Helma, C., King, R. D., Kramer, S., and Srinivasan, A. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001. http://www.predictive-toxicology.org/ptc/.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *CoRR*, 2020.

Isella, L., Stehlé, J., Barrat, A., Cattuto, C., Pinton, J.-F., and Van den Broeck, W. What's in a crowd? Analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166–180, 2011.

Jin, W., Barzilay, R., and Jaakkola, T. S. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pp. 2328–2337, 2018.

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., Zaslavsky, L., Zhang, J., and Bolton, E. E. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, 10 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representation*, 2017.

Kireev, D. B. Chemnet: A novel neural network based method for graph/property mapping. *Journal of Chemical Information and Computer Sciences*, 35(2):175–180, 1995.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

Knyazev, B., Taylor, G. W., and Amer, M. R. Understanding attention and generalization in graph neural networks. In *Neural Information Processing Systems*, pp. 4204–4214, 2019.

Kondor, R. and Pan, H. The multiscale Laplacian graph kernel. In *Advances in Neural Information Processing Systems*, pp. 2982–2990, 2016.

Kriege, N. M., Giscard, P.-L., and Wilson, R. C. On valid optimal assignment kernels and applications to graph classification. In *Advances in Neural Information Processing Systems*, pp. 1615–1623, 2016.

Kriege, N. M., Fey, M., Fisseler, D., Mutzel, P., and Weichert, F. Recognizing cuneiform signs using graph based methods. In *The International Workshop on Cost-Sensitive Learning*, 2018a.

Kriege, N. M., Morris, C., Rey, A., and Sohler, C. A property testing framework for the theoretical expressivity of graph kernels. In *International Joint Conference on Artificial Intelligence*, pp. 2348–2354, 2018b.

Kriege, N. M., Neumann, M., Morris, C., Kersting, K., and Mutzel, P. A unifying view of explicit and implicit feature maps of graph kernels. *Data Minining and Knowledge Discovery*, 33(6):1505–1547, 2019.

Kriege, N. M., Johansson, F. D., and Morris, C. A survey on graph kernels. *Applied Network Science*, 5(1):6, 2020.

Merkwirth, C. and Lengauer, T. Automatic generation of complementary descriptors with molecular graph networks. *Journal of Chemical Information and Modeling*, 45(5): 1159–1168, 2005.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5425–5434, 2017.

Morris, C., Kriege, N. M., Kersting, K., and Mutzel, P. Faster kernel for graphs with continuous attributes via hashing. In *IEEE International Conference on Data Mining*, pp. 1095–1100. IEEE, 2016.

Morris, C., Kersting, K., and Mutzel, P. Glocalized Weisfeiler-Lehman kernels: Global-local feature maps of graphs. In *IEEE International Conference on Data Mining*, pp. 327–336, 2017.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: Efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. Matching node embeddings for graph similarity. In *AAAI Conference on Artificial Intelligence*, pp. 2429–2435, 2017.

Nikolentzos, G., Meladianos, P., Limnios, S., and Vazirgiannis, M. A degeneracy framework for graph similarity. In *International Joint Conference on Artificial Intelligence*, pp. 2595–2601, 2018.

Oettershagen, L., Kriege, N. M., Morris, C., and Mutzel, P. Temporal graph kernels for classifying dissemination processes. *CoRR*, abs/1911.05496, 2019.

Ramakrishnan, R., Dral, P., O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Rieck, B., Bock, C., and Borgwardt, K. M. A persistent Weisfeiler-Lehman procedure for graph classification. In *International Conference on Machine Learning*, pp. 5448–5458, 2019.

Riesen, K. and Bunke, H. Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop*, pp. 287–297, 2008.

Rozemberczki, B., Kiss, O., and Sarkar, R. An API oriented open-source python framework for unsupervised learning on graphs. *CoRR*, abs/2003.04819, 2020.

Rozenshtein, P., Gionis, A., Prakash, B. A., and Vreeken, J. Reconstructing an epidemic over time. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1835–1844, 2016.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., and Schomburg, D. Brenda, the enzyme database: updates and major new developments. *Nucleic Acids Research*, 32(Database-Issue):431–433, 2004.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T. H., Mehlhorn, K., and Borgwardt, K. M. Efficient graphlet kernels for large graph comparison. In *International Conference on Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12: 2539–2561, 2011.

Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., and Vazirgiannis, M. Grakel: A graph kernel library in python. *CoRR*, abs/1806.02193, 2018.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 29–38, 2017.

Sperduti, A. and Starita, A. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(2):714–35, 1997.

Stokes, J., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N., MacNair, C., French, S., Carfrae, L., Bloom-Ackerman, Z., Tran, V., Chiappino-Pepe, A., Badran, A., Andrews, I., Chory, E., Church, G., Brown, E., Jaakkola, T., Barzilay, R., and Collins, J. A deep learning approach to antibiotic discovery. *Cell*, 180:688–702.e13, 02 2020.

Sugiyama, M. and Borgwardt, K. M. Halting in random walk kernels. In *Advances in Neural Information Processing Systems*, pp. 1639–1647, 2015.

Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., and Borgwardt, K. M. Wasserstein Weisfeiler-Lehman graph kernels. In *Advances in Neural Information Processing Systems*, pp. 6436–6446, 2019.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations*, 2016.

Viswanath, B., Mislove, A., Cha, M., and Gummadi, K. P. On the evolution of user interaction in facebook. In *ACM Workshop on Online Social Networks*, pp. 37–42, 2009.

Wale, N., Watson, I. A., and Karypis, G. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3): 347–375, 2008.

Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Ma, C., Huang, Z., Guo, Q., Zhang, H., Lin, H., Zhao, J., Li, J., Smola, A. J., and Zhang, Z. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/abs/1909.01315.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yan, X., Cheng, H., Han, J., and Yu, P. S. Mining significant graph patterns by leap search. In *ACM SIGMOD International Conference on Management of Data*, pp. 433–444, 2008.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *ACM SIGKDD International Conference on Knowledge Discovery and Data*, pp. 1365–1374. ACM, 2015.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

A. Evaluation examples

Kernelized SVM for graph kernels based on Gram matrices

```
import auxiliarymethods.auxiliary_methods as aux
import auxiliarymethods.datasets as dp
import kernel_baselines as kb
from auxiliarymethods.kernel_evaluation import kernel_svm_evaluation

# Download dataset.
classes = dp.get_dataset("ENZYMES")
use_labels, use_edge_labels = True, False

all_matrices = []
# Compute 1-WL kernel for 1 to 5 iterations.
for i in range(1, 6):
    # Use node labels and no edge labels.
    gm = kb.compute_wl_1_dense("ENZYMES", i, use_labels, use_edge_labels)
    # Apply cosine normalization.
    gm = aux.normalize_gram_matrix(gm)
    all_matrices.append(gm)

# Perform 10 repetitions of 10-CV using LIBSVM.
print(kernel_svm_evaluation(all_matrices, classes,
                            num_repetitions=10, all_std=True))
```

Linear SVM for graph kernels based on sparse feature maps

```
import auxiliarymethods.auxiliary_methods as aux
import auxiliarymethods.datasets as dp
import kernel_baselines as kb
from auxiliarymethods.kernel_evaluation import linear_svm_evaluation

# Download datasets.
classes = dp.get_dataset("MOLT-4")
use_labels, use_edge_labels = True, True

all_matrices = []
# Compute 1-WL kernel for 1 to 5 iterations.
for i in range(1, 6):
    # Use node labels and edge labels.
    gm = kb.compute_wl_1_sparse(dataset, i, use_labels, use_edge_labels)
    # Apply \ell_2 normalization.
    gm_n = aux.normalize_feature_vector(gm)
    all_matrices.append(gm_n)

# Perform 10 repetitions of 10-CV using LIBINEAR.
print(linear_svm_evaluation(all_matrices, classes,
                            num_repetitions=10, all_std=True))
```

GNN evaluation

```
import auxiliarymethods.datasets as dp
from auxiliarymethods.gnn_evaluation import gnn_evaluation
from gnn_baselines.gnn_architectures import GIN
from auxiliarymethods.reader import tud_to_networkx

dataset = "PROTEINS"
use_labels = True

# Download datasets.
dp.get_dataset(dataset)

# Optimize the number of layers ({1,2,3,4,5}) and
# the number of hidden features ({32,64,128}),
# set the maximum nummber of epochs to 200,
```

```
# batch size to 64,
# start learning rate to 0.01, and
# number of repetitions for 10-CV to 10.
print(gnn_evaluation(GIN, dataset, [1, 2, 3, 4, 5], [32, 64, 128], max_num_epochs=200,
                     batch_size=64, start_lr=0.01, num_repetitions=10, all_std=True))
```

Loading graphs as NetworkX graphs

```
import auxiliarymethods.datasets as dp
from auxiliarymethods.gnn_evaluation import gnn_evaluation

dataset = "PROTEINS"

# Download datasets.
dp.get_dataset(dataset)
# Output datasets as a list of graphs.
graph_db = tud_to_networkx(dataset)
```

B. Experimental protocol and hyperparameters for ZINC, ALCHEMY, QM9

For the larger molecular regression tasks, ZINC and ALCHEMY,[5] we closely followed the hyperparameters found in (Dwivedi et al., 2020) and (Chen et al., 2019a), respectively, for the GINE-ε layers. That is, for ZINC, we used four GINE-ε layers with a hidden dimension of 256 followed by batch norm and a 4-layer MLP for the joint regression of the twelve targets, after applying mean pooling. For ALCHEMY and QM9, we used six layers with 64 (hidden) node features and a set2seq layer (Vinyals et al., 2016) for graph-level pooling, followed by a 2-layer MLP for the joint regression of the twelve targets.

For ZINC, we used the given train, validation split, test split, and report the MAE over the test set. For the ALCHEMY and QM9 datasets, we uniformly and at random sampled 80% of the graphs for training, and 10% for validation and testing, respectively. Moreover, following (Chen et al., 2019a; Gilmer et al., 2017), we normalized the targets of the training split to zero mean and unit variance. We used a single model to predict all targets. Following (Klicpera et al., 2020), we report mean standardized MAE and mean standardized logMAE. We repeated each experiment five times (with different random splits in case of ALCHEMY and QM9) and report average scores and standard deviations. Moreover, we use a learning rate decay of 0.5 with a patience parameter of 5, and a starting learning rate of 0.001 with a minimum of 10^{-6}.

For the QM9 dataset, we additionally used the MPNN (Gilmer et al., 2017) architecture as a baseline, closely following the setup of (Gilmer et al., 2017). For the GINE-ε and the MPNN architecture, following Gilmer et al. (Gilmer et al., 2017), we used a complete graph, computed pairwise ℓ_2 distances based on the 3D-coordinates, and concatenated them to the edge features. We note here that our intent is not to beat the state-of-the-art, physical knowledge-incorporating architectures, e.g., DIMENET (Klicpera et al., 2020) or CORMORANT (Anderson et al., 2019), but to solely provide baseline scores.

All neural experiments were conducted on a workstation with four Nvidia Tesla V100 GPU cards with 32GB of GPU memory running Oracle Linux Server 7.7.

C. Dataset statistics

[5] Note that the full dataset is different from the contest dataset, e.g., it does not provide normalized targets, see https://alchemy.tencent.com/.

Table 4. Dataset statistics and properties, †—Continuous vertex labels following (Gilmer et al., 2017), the last three components encode 3D coordinates.

Dataset	**Properties**					
	Number of graphs	Number of classes/targets	∅ Number of vertices	∅ Number of edges	Vertex labels	Edge labels
ENZYMES	600	6	32.6	62.1	✓	✗
IMDB-BINARY	1 000	2	19.8	96.5	✗	✗
IMDB-MULTI	1 500	3	13.0	65.9	✗	✗
NCI1	4 110	2	29.9	32.3	✓	✗
NCI109	4 127	2	29.7	32.1	✓	✗
PTC_FM	349	2	14.1	14.5	✓	✗
PROTEINS	1 113	2	39.1	72.8	✓	✗
REDDIT-BINARY	2 000	2	429.6	497.8	✗	✗
YEAST	79 601	2	21.5	22.8	✓	✓
YEASTH	79 601	2	39.4	40.7	✓	✓
UACC257	39 988	2	26.1	28.1	✓	✓
UACC257H	39 988	2	46.7	48.7	✓	✓
OVCAR-8	40 516	2	26.1	28.1	✓	✓
OVCAR-8H	40 516	2	46.7	48.7	✓	✓
ZINC	249 456	12	23.1	24.9	✓	✓
ALCHEMY	202 579	12	10.1	10.4	✓	✓
QM9	129 433	12	18.0	18.6	✓(13+3D)†	✓(4)