QUADRATIC GCN FOR GRAPH CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Convolutional Networks (GCN) have been extensively used to classify nodes in graphs and have been shown to outperform most other node classification methods. However, there are currently few GCN based formalism for graph classification tasks. In this task, graphs of different sizes (e.g. graphs representing different protein structures) belong to different classes, and one attempts to predict the graph class.

We here propose a solution combing GCN and methods from knowledge graphs to produce a quadratic GCN (Q-GCN). We extend the GCN formalism by adding a quadratic layer to a standard GCN to classify full graphs. Such a layer produces an output with dimensions independent of the graph node number. This output is then passed through a softmax to classify full graphs. We applied this method to a wide range of graph classification problems and show that such a straightforward formalism outperforms state of the art methods for binary graph classification with or without external input on each graph.

Keywords - GCN, Quadratic activation, graph classification

1 INTRODUCTION

Information is often represented through relations between entities. A simple formalism to represent such relations are graphs relating nodes by edges that can be directed and weighted. Given such graphs, a classification task is often required. In most studies, the task was the classification of the nodes of a given graph. For example, given the citations between papers, one could attempt to classify the paper domain (Lu & Getoor, 2003). Such tasks are often performed through information diffusion in the network, following the assumption that neighboring nodes have similar classes (Bhagat et al., 2011; Tang & Liu, 2011; Angelova & Weikum, 2006). Recently more complex approaches have been proposed that use the network topology (Rosen & Louzoun, 2015; Naaman et al., 2018; Grover & Leskovec, 2016) or the propagation of external information (Kato et al., 2009; Speriosu et al., 2011). A dominant approach for such classifications (either supervised or semi-supervised) is Graph Convolutional Networks (GCN) (Kipf & Welling, 2016). GCNs are based on weighting a signal propagated through the network. The general formalism of a layer in GCNs is:

$$X_k = \sigma(X_{k-1} * \tilde{A} * W_k), \tag{1}$$

where A is a matrix derived from the adjacency matrix. For example, a symmetrizing or normalized adjacency matrix:

$$\tilde{A} = D^{-1/2}[A + A^T + I]D^{-1/2}. \tag{2}$$

W_k and X_k are weights and input values of the k^{th} layer, where the 0 layer is the input layer. The output layer has dimensions of the node number by the number of classes and can be passed through a softmax to produce a classification of each node. Over the last 3 years over 1,500 extensions and applications of GCN have been published in combination with many other learning methods, including, among many others, combinations of GCN with recurrent neural networks (Ling et al., 2019), with GANs (Lei et al., 2019) and with active learning (Abel & Louzoun, 2019).

A related task, much less studied, is the classification of full graphs of different sizes. For example, proteins can be classified based on their molecular interaction graphs. Organizations can be grouped based on their communication graphs. In such a case a single label is required for each graph. This poses a challenge to GCN based approaches since those produce a projection for each node. As such their output dimension is determined by the number of nodes. Current approaches for solving the network size are indirect and are based on statistical properties of the node projection or the network itself (Pan et al., 2015; 2016), or pooling (Zhang et al., 2018).

We here propose a direct approach, adopted from knowledge graphs (Hamilton et al., 2017), which is to add a quadratic layer to the GCN :

$$X_{end} = \sigma(V_1^T * X_{k1}^T * \tilde{A} * X_k * V_2), \tag{3}$$

where V_1 and V_2 are different weight matrices, and X_k and X_{k1} are the input values to the k layer. $X_k 1$ can be either the original input or the output of any layer. Such a formalism outputs a result of fixed dimensions equal to the external dimensions of V_1 and V_2. We here propose that using such a formalism with an output size of the number of classes by 1 can be directly used to classify networks, and denote this formalism Quadratic GCN (Q-GCN). The Q-GCN produces for each network a softmax output on each of the possible classes for this network. This output is then compared to the true label to learn the classification task.

2 Relation to previous work

Graph classification task have emerged in three independent domains: Chemoinformatics, graph kernel analysis, and GCN.

In Cheminformatics, the function of molecules is predicted from their structure and composition. In such a task, each molecule is translated to a network of atoms or amino acids, and each node is associated with biochemical properties, such as its molecular weight, charge. etc. Edges can also be associated with the physical distance between the nodes, or the type of molecular bond. Many of the standard datasets for graph class inference come from this domain. We have here tested three such examples, including the classification of protein classes (Borgwardt et al., 2005b), the classification of mutagens (Kazius et al., 2005) and the classification of molecules that can be used as drugs (Gaüzere et al., 2012).

Beyond chemoinformatics, graph comparison has been traditionally performed using kernel approaches. Graph kernels exploit graph topology but restrict themselves to comparing substructures of graphs that are computable in polynomial time. Many graph kernels have been defined, which focus on different types of substructures in graphs, such as random walks (Kashima et al., 2004; Gärtner et al., 2003) , shortest paths (Aßfalg et al., 2006), limited size graphlets (also denoted motifs) (Shervashidze et al., 2011) and subtrees (Ramon & Gärtner, 2003). Note that once a distance kernel between graphs has been defined, classification itself is straightforward using distance-based classification methods.

GCNs have been also extended to graph classification. The main efforts in this direction were DIFFPOOL, a differentiable graph pooling module that can generate hierarchical representations of graphs and used this hierarchy of node groups to classify graphs (Ying et al., 2018). Zhang et al. (Zhang et al., 2018) have used a formalism very similar to the Kipf and Welling formalism but added a pooling strategy to the last layer with a SortPooling layer. Pan et al. have produced a couple of algorithms of Multi Task Learning (MTL). MTL jointly discovers discriminative subgraph features across different tasks and uses them to classify graphs (Pan et al., 2015; 2016). As such, their work may be seen as an extension of kernel methods.

We here propose to adapt from knowledge graphs quadratic layers and merge them with GCN to classify graphs. A knowledge graph (KG) is a multi-digraph with labeled edges, where the label represents the type of the relationship. Such graphs have been used in multiple graph classification tasks. A general framework for KG is first an encoder to project the nodes, and then a decoder that compresses node representations into a single representation. Then, to classify full graphs, Duvenaud et al. (Duvenaud et al., 2015) averaged all the node projections. Gilmer et al. (Gilmer et al., 2017) used a DeepSet aggregation. Li et al. (Li et al., 2015) added a new virtual node and connected it to all the nodes. They then used the representation of the virtual node as the representation of the graph. However, our interest in KG is in the bilinear decoders proposed for link predictions Wang et al. (2018). We here propose to transfer bilinear decoders and merge them with GCN to produce graph classifiers.

3 Main Claims

- We propose a "natural" formalism for graph classification that is a direct extension of GCN. This formalism is a GCN on all but the last layer and a bilinear GCN layer in the last layer. This formalism produces a fixed size output for graphs of different sizes.
- We show that this formalism can outperform all current methods for graph classification in a couple of previously studied binary classification tasks.
- We propose new methods to classify graphs with no external information (only the adjacency matrix) using topological features of each node as an input to each node in the GCN.

4 Model

Each node is assigned an input vector $x_0(i)$ defined to be either some external information on the node or topological features of the node. The values of the following layers are computed as in Eq. 1 and 2, except for the last layer, which is computed as in Eq. 3, with a similar matrix and a softmax nonlinearity. The loss function is then defined as the cross-entropy of the last layer with the network classification. We train the weights to minimize the loss function over all studied networks. The formalism can be formalized as:

$$\begin{aligned} \tilde{A} &= D^{-0.5} * [A + A^T + I] * D^{-0.5} \\ X_k &= \sigma(X_{k-1} * \tilde{A} * W_k) \\ X_{end} &= \sigma(V_1^T * X_{k-1}^T * \tilde{A} * X_k * V_2). \end{aligned} \tag{4}$$

This formalism translates each graph into a constant size output (Figure 1). As such, it is not sensitive to the graph size. The output is directly amenable to standard loss minimization for graph classification problems by comparing the fixed size output with the expected classification of the graph.

The model's input consists of three parameters.

- The first is the normalized adjacency matrix of the graph$\tilde{A}$.
- The second is the input matrix $x_0 \in R^{n*d}$ were n is the number of nodes in the graph. x_0 is a concatenation of topological features and external features for each node: The topological features used were: In/Out degree (or just degree if the graph is undirected), Centrality and first and second moments of the distance distribution of each node to all other nodes in the graph.
- The third parameter is an embedding vector. Some of the external features are given as classes and by representing classes as one hot vectors, valuable information is lost. To learn representation that will reflect similarity between classes, a use of embedding is required. The embedding dimension used was 20 for each external feature.

All input features were z-scored. The non-linearity of the GCN layers was a Relu, and the non-linearity for the quadratic layer was a sigmoid for the binary cases, and a softmax otherwise.

We here used 2 layers with output dimensions of 200 each. The QGCN then takes the adjacency matrix x_0 and x_1, and pass them as an input for a single bilinear layer to produce the final result. The learning rate was $1e-4$, and an ADAM optimizer was used. The regularization was an L2-regularization with a coefficient of $2e-5$ and the loss function was Cross-Entropy.

Two types of experiments were performed. In the first experiment, the data-sets were shuffled and split according to the split ratio of the compared experiments. Then using NNI (https://github.com/Microsoft/nni), a broad hyper-parameter tuning was applied, optimized based on the internal validation. The NNI was used in two steps. First, a grid search of a wide range of parameters was performed to get the amplitude of the regularization. The second step was to refine the outcome by setting the tuner to Tree-structure Paezen Estimator (TPE) and running another search. The second experiment tested the model success as a function of train size. at this case the dev-set was fixed to be 5% of the data, while the test set was between $5\% - 85\%$.

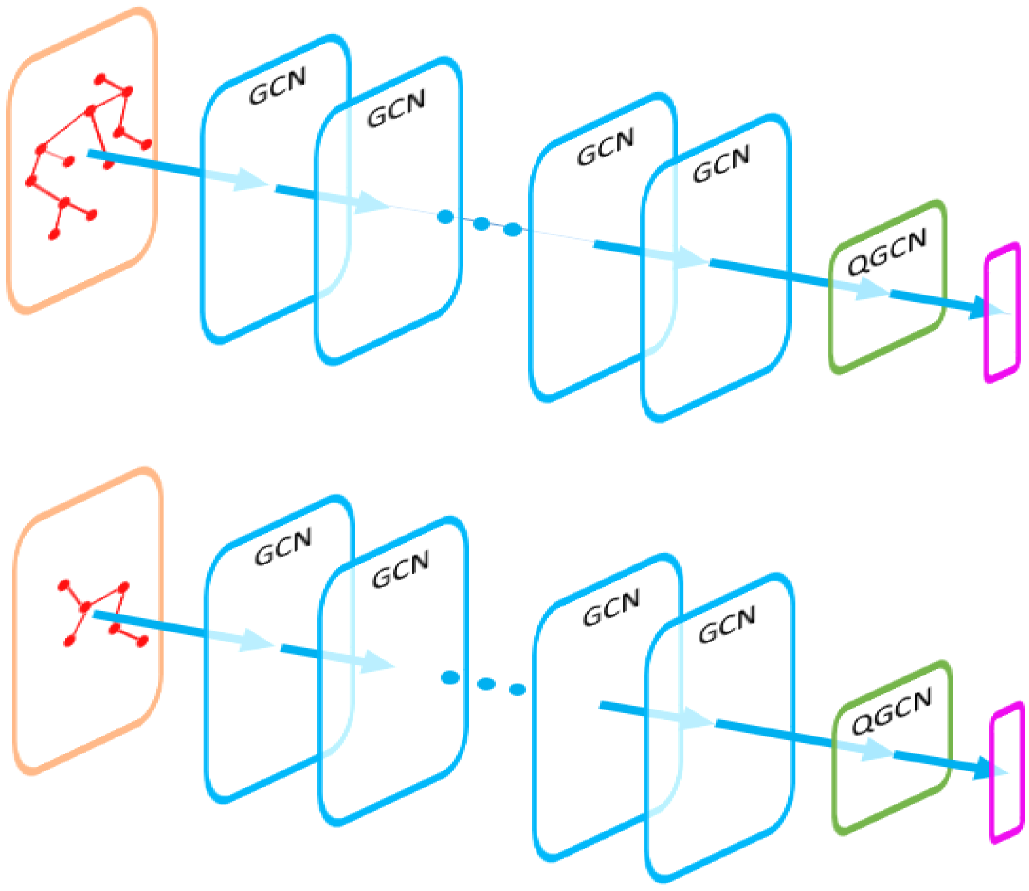

Figure 1: *Model description. Graphs of different sizes are passed through GCN layers. The output size of each GCN layer is proportional to the number of nodes in the current graph. The last layer is a QGCN which gets as input which is proportional to the graph number of nodes and outputs a constant size output. Note that the weights of the QGCN are of fixed dimensions and as such can be learned on all graphs simultaneously.*

4.1 Datasets studied

We studied multiple datasets that were classified with previous methods:

- HIV related molecules (Gaüzere et al., 2012). This dataset consists of graphs representing molecular compounds. Each compound can be classified as active or inactive. The labels indicate if a molecular compound is active against HIV or not. The nodes and the edges of the graphs represent atoms and their covalent bounds respectively. The dataset also contains non-topological information about the nodes of the graph such as the chemical compound and the charge of the atom.
- Mutagens (Kazius et al., 2005) . As with the previous dataset, this dataset consists of graphs representing molecules. The compounds are converted to graphs in the same way. Each graph is labeled as mutagen or non-mutagen. A mutagen chemical induces mutations in the genetic material of an organism and can reduce the potential of compound to become a marketable drug.
- Protein classification Dataset (Borgwardt et al., 2005b). This dataset consists of graphs representing proteins constructed from the Protein Data Bank and labeled with their six corresponding enzyme class labels from the Brenda enzyme database, which determine their functionality. The nodes of the graphs represent amino acids and every node is connected to its three nearest neighbors in space.
- Grec drawing classification. (Nene et al., 1996). The dataset consists of graphs that were produced by symbols from architectural and electronics drawings. The images occur in five different distortion levels and the graphs are extracted from the resulting de-noised images by tracing the lines from end to end and detecting intersections as well as corners. the dataset contains 1,100 graphs uniformly distributed over 22 classes.

5 Results

5.1 Comparison with state of the art

We tested the accuracies of the Q-GCN on the datasets above and compared it to the best published accuracies (dashed line). We have performed for each training set fraction 10 learning attempts (for the same training/test division), and used the one providing the lowest loss on the training set. The train/test split was different for each experiment and defined following the experiment compared to. In two of the four datasets, we obtained a higher accuracy than previously published. In the AIDS reaction dataset, previous accuracy was above 99.5 %, and here we obtain a perfect classification. In the mutagen dataset, we obtain a significant advance compared with the state of the art (Table 1). Note that while in binary classification tasks, the Q-GCN provides higher accuracy than the current state of the art, in multi-class tasks, the Q-GCN has significantly lower performance than current state of the art (Table 1). This may be the result of sensitivity to hyper-parameters of the multi-class problem or an inherent weakness of Q-GCN.

Also, although the Q-GCN is over-parameterized and the regularization is weak, the overfitting is very limited.For fixed hyper-parameters and fixed train/test split, the obtained loss and accuracy over the train test, vary as a function of the random initial condition. The test accuracy follows the training set accuracy (Fig 2).

5.2 Sensitivity analysis

As is the case in all GCN formalisms, the network expects some input for each node to train the GCN. Often external information is used when available. A typical example would be the Bag Of Words (BOW) representation of manuscripts used in the CORA classification task. In the absence of external information, Kipf et al. (Kipf & Welling, 2016) have recently proposed to use an identity matrix as input, representing the id of each node through a one hot formalism. More recently, Benami et al. (2019) have proposed using topological features, such as the degree, or the centrality of each node . We have tested the possibility of using a combination of external information with topological features as an input. We have further tested the effect of changing the layer multiplying

Table 1: Results

Dataset	Method	Acc.	AUC
H.I.V.	Neuhaus & Bunke (2007) *	**0.997**	—
H.I.V.	Q-GCN	0.983	0.977
Mutagenicity	Kazius, J. & McGuire, R. & Bursi, R. (2005) **	0.715	—
Mutagenicity	Q-GCN	**0.726**	0.748
Protein	Karsten, M. ***	**0.655*****	—
Protein	Q-GCN	0.316	—
Grec	Dosch, Ph. and Valveny, E. ****	**0.955**	—
Grec	Q-GCN	0.609	—

* book - bridging the gap between graph edit distance and kernel machine
** Protein function prediction via graph kernels
*** as reported in (iapr-tc15.greyc.fr) citing Borgwardt et al. (2005a). However, the report in is only for binary comparisons.
**** book - Graphics Recognition. Ten years of review and future perspective

the bilinear term to be either the first or the second layer. Finally, we have tested whether a normalized (Eq. 2) or non-normalized $\tilde{A} = [A + A^T + I]$ matrix should be used in each layer. Consistently and over all data sets and all training fraction, using the input layer provides higher accuracies than the second layer similarly and using a non-normalized matrix provides higher accuracies than a normalized one. (Fig 3).

6 Discussion

Methods to compare graphs have been developed in many domains, including chemoinformatics (Kashima et al., 2004; Gärtner et al., 2003), sub-graph isomorphism (Shervashidze et al., 2009), graph classification tasks (Li et al., 2015) and many others. Many of these methods were kernel-based, where properties of the graph were defined and graphs were compared based on these properties similarity. A simple example would be counting the frequency of specific subgraphs. An alternative approach was to project the graph and compare properties of the projection. One could compare for example the moments of the distribution. Generally, all such methods define a distance matrix between graph properties.Once such a distance is defined, supervised and unsupervised distance-based approaches can be applied to a set of graphs.

However, such approaches require a predefinition of the features to be compared. As such, these methods may require adaptation to different types of graphs that may have different properties. Instead, we have here proposed an alternative method based on the extension of GCN. Recently, GCNs have been developed to classify nodes within a graph and have revolutionized node classification tasks (Kipf & Welling, 2016). GCN can be viewed as a projection of nodes into a real matrix whose dimensions are the number of nodes and a constant representation dimension. The extension of GCN to graph classification has been proposed based on either pooling methods or through hierarchical approaches.

The Q-GCN proposed here is a direct extension of GCN to a quadratic format (QGCN), which is naturally amenable to graph classification tasks, and as such can be seen as the direct extension of GCN to graph classification tasks. Instead of predefining the properties of the projection to best classify graphs, these properties are learned through a combination of weight multiplying different

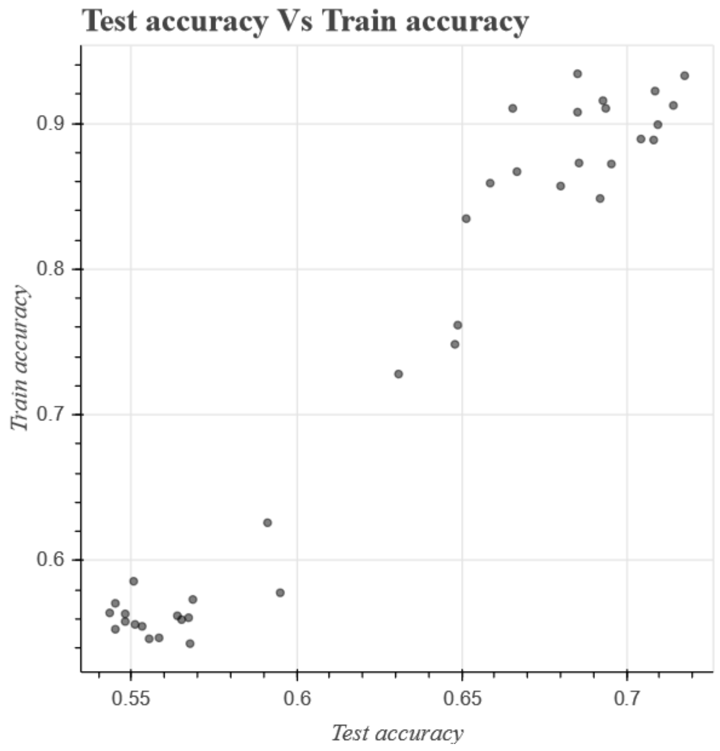

Figure 2: *Test and train accuracy. We ran 40 experiments, applying the Q-GCN on the Mutagenicity dataset. The x-axis shows the test accuracy, the y-axis shows the train accuracy and each dot represents a single experiment.*

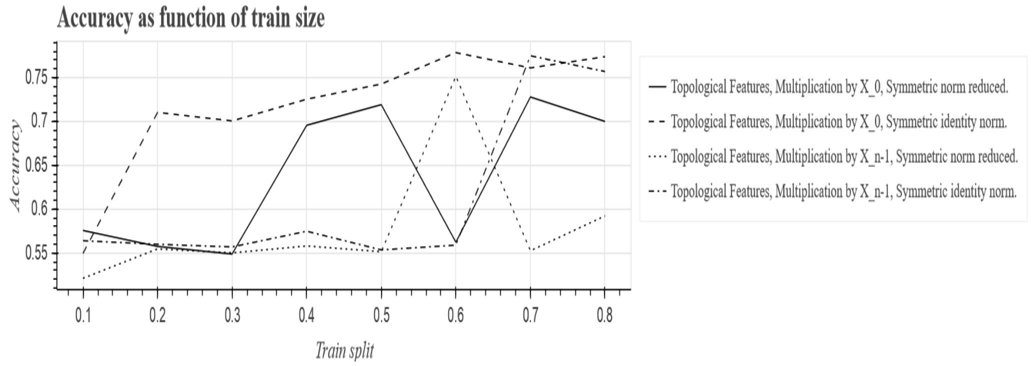

Figure 3: *Model comparison. We compared the 4 possible models on the Mutagenicity dataset. Either normalized or non-normalized adjacency matrix, and either first or second layer as the second component in bilinear term. The y-axis is the accuracy of the test set and the x-axis is the training set fraction.*

projections of the graphs using a quadratic product. Such products have been previously proposed in multiple contexts, but mainly knowledge graphs. We have here tested the application of Q-GCN to categorical classification tasks (either binary or multi-class) and shown that in multiple datasets it outperforms all existing approaches. Its extension to regression problems or semi-supervised approaches is straightforward. Interestingly, while in binary classification, the Q-GCN achieves very high accuracies, in multi-class tasks, it does not reach the accuracy of state of the art methods. A possible explanation may be that different labels may need significantly different weighting that cannot be learned within a single learning framework. Another interesting result is that using the input layer as an input to the quadratic layer produces higher accuracy than using any other layers' output. We have currently no clear explanation for that. Note that the input layer used here is a combination of topological features as proposed by Benami et al. (2019). This may represent the importance of topological features for node classification.

The advantages of QGCN beyond its accuracy are its simplicity, its generic structure and the resulting possible application to many domains almost blindly. A possible problem with QGCN is the numerical failure to converge in some cases.In all the datasets studied here, some of the optimizations did not converge to low loss function values on the training set. However, choosing the results with the optimal training set loss function always led to high enough accuracies. Still, these stability issues should be studied to produce an optimization technique adapted to such quadratic functions. We now plan on developing such optimization methods.

REFERENCES

Roy Abel and Yoram Louzoun. Regional based query in graph active learning. *arXiv preprint arXiv:1906.08541*, 2019.

Ralitsa Angelova and Gerhard Weikum. Graph-based text classification: learn from your neighbors. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 485–492. ACM, 2006.

Johannes Aßfalg, Karsten M Borgwardt, and Hans-Peter Kriegel. 3dstring: a feature string kernel for 3d object classification on voxelized data. In *Proceedings of the 15th ACM international Conference on information and Knowledge Management*, pp. 198–207. ACM, 2006.

Idan Benami, Keren Cohen, Oved Nagar, and Yoram Louzoun. Topological based classification of paper domains using graph convolutional networks. *arXiv preprint arXiv:1904.07787*, 2019.

Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pp. 115–148. Springer, 2011.

K. Borgwardt, C. Ong, S. Schönauer, S. Vishwanathan, A. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(1):47–56, 2005a.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005b.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Benoit Gaüzere, Luc Brun, and Didier Villemin. Two new graphs kernels in chemoinformatics. *Pattern Recognition Letters*, 33(15):2038–2047, 2012.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Kernels for graphs. *Kernel methods in computational biology*, 39(1):101–113, 2004.

Tsuyoshi Kato, Hisahi Kashima, and Masashi Sugiyama. Robust label propagation on multiple networks. *IEEE Transactions on Neural Networks*, 20(1):35–44, 2009.

Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1):312–320, 2005.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 388–396. IEEE, 2019.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5257–5266, 2019.

Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 496–503, 2003.

Roi Naaman, Keren Cohen, and Yoram Louzoun. Edge sign prediction based on a combination of network structural topology and sign propagation. *Journal of Complex Networks*, 7(1):54–66, 2018.

S. Nene, S. Nayar, and H. Murase. Columbia Object Image Library: COIL-100. Technical report, Department of Computer Science, Columbia University, New York, 1996.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and S Yu Philip. Joint structure feature exploration and regularization for multi-task graph classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):715–728, 2015.

Shirui Pan, Jia Wu, Xingquan Zhu, Guodong Long, and Chengqi Zhang. Task sensitive feature exploration and learning for multitask graph classification. *IEEE transactions on cybernetics*, 47 (3):744–758, 2016.

Jan Ramon and Thomas Gärtner. Expressivity versus efficiency of graph kernels. In *First international workshop on mining graphs, trees and sequences*, pp. 65–74. Citeseer, 2003.

Yonatan Rosen and Yoram Louzoun. Topological similarity as a proxy to content similarity. *Journal of Complex Networks*, 4(1):38–60, 2015.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pp. 53–63. Association for Computational Linguistics, 2011.

Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

Yanjie Wang, Rainer Gemulla, and Hui Li. On multi-relational link prediction with bilinear models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.