https://dx.doi.org/10.22161/ijaers.6.1.11 ISSN: 2349-6495(P) | 2456-1908(O)

The Extension of Graph Convolutional Neural Network with Capsule Network for Graph Classification

Jean de DIEU TUGIRIMANA[1*], Janvier RULINDA[2], Antoine NZARAMBA[3]

[1,2]Master's Student in computer science at Central South University, Changsha, China.
[3]PhD Student in computer science at Central South University, Changsha, China.
* Corresponding author. Tel.: +8615773172105; email: titijado@gmail.com

Abstract*— *In this paper we extend a graph convolutional neural network (GCNNs) which is the one of the existing state-of-art deep learning methods using the notion of capsule networks for graph classification. Through experiments, we show that by extending GCNNs using capsule networks can significantly overcome the challenges of GCNNs for the task of graph classification.
Keywords*— *Capsule Network, Graph Convolutional neural networks.

I. INTRODUCTION

Many real-world problems are represented as graphs, such as social networks, molecular graph structures, biological protein-protein networks, all of these domains and many more can be readily modeled as graphs, which capture interactions (i.e., edges) between individual units(i.e., nodes).

In many of these problems, the input data is in the form of graphs, and the graph convolutional neural networks task is to do node classification and graph classification. Graph classification, or the problem of identification of class labels of graphs in a dataset, is an important problem with practical applications in a diverse set of fields. Data from bioinformatics [1], chemoinformatics [2], social network analysis[3], urban computing[4], and cyber-security [5] can all be naturally represented as labeled graphs.

The standard graph convolutional neural networks model commonly used in existing deep learning approaches on graphs, especially when it applied to the graph classification problem it face some limitations :

- Loss of information due to the basic graph convolution operation.
- Graph convolutional neural network model are equivariant because of this it cannot apply directly to graph classification problem, since it cannot provide any guarantee that the outputs of any two isomorphic graph graphs are always the same.
- Graph convolutional neural networks model are limited to exploiting global information for the purpose of graph classification

II. RELATED WORK

Many different techniques have been proposed to solve the graph classification problem. One popular approach is to use a graphkernel to measure similarity between different graphs [6]. This similarity can be measured by considering various structural properties like the shortest paths between nodes [7], the occurrence of certain graphlets or subgraphs [8], and even the structure of the graph at different scales [9].

Recently, several new methods which generalize over previous approaches have been introduced. These methods use a deep learning framework to learn data-driven representations of graphs [10, 11, 12]. In [10], a method is introduced that generalizes the Weisfeiler-Lehman (WL) algorithm by learning to encode only relevant features from a node'sneighborhood during each iteration. Interestingly, [11] proposes a method that processes a section of the input graph using a convolutional neural network. However, for this to work for graphs of arbitrary sizes the method relies on a labeling step that ranks all the nodes in the graph which means it still processes the entire graph initially

III. PROPOSED MODEL AND CONTRIBUTION

The main contributions of our paper can be summarized as follows:

1. Proposing a novel Graph Convolution neural network with Capsule Networks (GCNN-CapsNet) model which is based on the capsule idea of capturing high information output in a small vector instead of scaler output which is current used on GCNN models.

https://dx.doi.org/10.22161/ijaers.6.1.11 *ISSN: 2349-6495(P) | 2456-1908(O)*

2. Replacing the max pooling and node aggregation which is current methods used to achieve the graph permutation invariance by developing a novel graph permutation invariant layer which is based on computing the covariance of the data to solve graph classification problem.
3. Designing GCNN-CapsNet which will exploit the global graph structure features at each graph node.

3.1 Proposed model

Our proposed model has Capsule networks as core idea behind is to capture more information in local node pool beyond what captured by max pooling and by aggregation, which graph convolution operation is used in a standard GCNN model. The new information is encapsulated in so called instantiation parameters described in [13] which form a capsule vector of highly informative output

3.2 GCNN general model

For the basic notations let consider a graph $G = (V, E, A)$ of size $N = |V|$ Where V is the vertex set, E the edge set and $A = [a_{ij}]$ the weighted adjacency matrix. The standard graph Laplacian is defined by $L = D - A \in \mathbb{R}^{NxN}$, where D is the degree matrix. Let $X \in \mathbb{R}^{Nxd}$ be the node feature matrix, where d is the input dimension.

Before describing our model GCNN-CapsNet let start by describing a general GCNN model. Let G be a graph with graph Laplacian Land $X \in \mathbb{R}^{Nxd}$ be a node feature matrix. then general form of a GCNN layer output function $f(X, L) \in \mathbb{R}^{Nxh}$ given by $f(X, L) = \delta(\sum_{k=0}^{K} L^K X W_K)$ (1)

Where $L^k X$ is graph convolution filter of polynomial form with degree k. while W_k are learning weight parameters.

3.3 Capsule graph function

Capsule graph function is described by considering an i^{th} node with x_0 value and the set of its neighborhood node values as $N(i) = \{x0, x1, x2..., xk\}$ including itself. In the standard graph convolution operation, the output is a scalar function $\mathbb{R}^k \rightarrow \mathbb{R}$ which take k input neighbors at the i^{th} node and yields an output given by $\mathrm{fi}(\mathrm{x0}, \mathrm{x1}, \mathrm{x2}, .., \mathrm{xk}) = \frac{1}{|\mathrm{N(i)}|} \sum_{\mathrm{k \in N(i)}} \mathrm{a_{ik}}\ \mathrm{x_k}$ (2)

Where a_{ik} represents edge between nodes i and k .

Our capsule graph network, we replace $f(x0, x1, \ldots., xk)$ with a vector valued capsule function $f: \mathbb{R}^k \rightarrow \mathbb{R}^p$. for example, consider a capsulea capsule function that capture higher order statistical moments as follows , we omit the mean and standard deviation for simplification

$$fi(xo, x1, \ldots, xn) = \frac{1}{|N(i)|} \begin{bmatrix} \sum_{k \in N(i)} a_{ik} x_k \\ \sum_{k \in N(i)} a_{ik} x_k^2 \\ . \\ . \\ . \\ \sum_{k \in N(i)} a_{ik} x_k^p \end{bmatrix} \quad (3)$$

3.4 Graph Capsule Vector Dimension

In the first layer of graph capsule network receives an input $X \in \mathbb{R}^{N*d}$ and produces a nonlinear output $f(X, L) \in \mathbb{R}^{N*h1*p}$. since the graph capsule function produce a vector of p dimension, the feature dimension of the output in subsequent layers can quickly blow up to an unmanageable value. For keeping checking, we restrict the feature dimension of the output $f^{(L)}(X, L)$ to be always $\in \mathbb{R}^{N*h1*p}$ at any middle l^{th} layer of GCNN-CapsNet. This was accomplished by flattening the last two dimension of $f(X, L)$ and carrying out graph convolution in usual way (for example see equation 4 for flattening).

3.5 Graph Capsule function with statistical moments

Considering higher-order statistical moments as instantiation parameters because they are permutation-ally invariant and can nicely be computed through matrix-multiplication operations in a fast manner. To do this let $f_p(X, L)$ be the output matrix corresponding to p^{th} dimension. Then we can compute $f_p^l(X, L)$ containing statistical moments as instantiation parameters as follows

$$f_p^{(l)}(X, L) = \delta(\sum_{k=0}^{k} L^k (f_F^{(l-1)}(X, L) \odot \ldots\ldots \odot f_F^{(l-1)}(X, L)) W_{pK}^{(l)} \quad (4)$$

Where $\odot$ is a hadamard product. Here to keep the feature dimension in check from growing, we flatten the last two dimension of the input as $f_{Flat}^{(l-1)}(X, L) \in \mathbb{R}^{N\ x\ h_l\ -1p}$

And perform usual graph convolution operation followed by a linear transformation with $W_{p^k}^{(l)} \in \mathbb{R}^{h_l\ x\ h_l\ -1p}$ as the learning weight parameter. Where p is used to denote both the capsule dimension as well the order of statistic moments.

3.6 Graph permutation invariant layer

The permutation invariant feature in GCNN-CapsNat model of computing the covariance of $f(X, L)$ layer output is given as follows,

$$C(f(X, L)) = \frac{1}{N} (f(X, L) - \mu)^T (f(X, L) - \mu) \quad (5)$$

Here μ is the mean of of $f(X, L)$ output and $C(.)$ is a covariance function. Since covariance function is

https://dx.doi.org/10.22161/ijaers.6.1.11 *ISSN: 2349-6495(P) | 2456-1908(O)*

differentiable and does not depend upon the order of row elements, it can serve as permutation invariant layer in GCNN-CapsNet model. And it is also fast in computation due to a single matrix-multiplication operation. Here we flatten the last two dimension of GCNN-CapsNet layer output of $f(X,L) \in \mathbb{R}^{Nxhxp}$ in order to compute the covariance.

In addition, covariance provides much richer information about the data by including shapes, norms and angles (between node hidden features) information rather than just providing the mean of data. In fact in multivariate normal distribution, it is used as a statistical parameter to approximate the normal density and thus also reflects information about the data distribution. This particular property along with invariance has been exploited before in [14] for computing similarity between two set of vectors. One can also think about fitting multivariate normal distribution onof $f(X,L)$ but it involves computing inverse of covariance matrix which is computationally expensive.

Since each element of covariance matrix is invariant to node orders, we can flatten the symmetric covariance matrix $C \in \mathbb{R}^{hp\,x\,hp}$ to construct the graph invariant feature vector $f \in \mathbb{R}^{(hp+1)hp/2}$. other positive note , here the output dimension of f does not depend upon N number of nodes and can be adjusted according to computational constraints.

It is quite straightforward to see that the feature dimension order of a node does not depend upon the graph node ordering and hence the order is same across all graphs. As a result, each element of $f1$ and $f2$ are always comparable. To be more specific, covariance output compares both the norms sand angles between the corresponding pairs of feature dimension vectors in two graphs.

3.7 GCNN-CapsNet Global Features

Another desired characteristic of graph classification problem is to capture global structure of graph. For instance, by considering only node degree (as a node feature) is a local information and is not much helpful towards solving graph classification problem. Also by considering spectral embedding as a node feature it takes global piece of information into account and has been proven successful in serving as a node vector for problems dealing with graph semi-supervised learning. We define a global feature that takes full graph structure into account during their computation. While local features only depend upon some k-hop node neighbors.

Unluckily, the basic design of GCNN model can only capture local structure information of the graph at each node.

Let G be a graph with $L \in \mathbb{R}^{N*N}$ graph laplacian and $X \in \mathbb{R}^{N*d}$ node feature matrix. let $f^{(l)}(X,L)$ be the output function of a l^{th} GCNN layer equipped with polynomial filters of degree k. Then $[f^{(l)}(X,L)]i$ output at i^{th} depends upon "only" on the input values of neighbors distant at most "kl-hops" away.

Mathematical we can proof the above statement, it is easy to see that the base case $l = 1$ holds true. Let's assume it also holds true for $f^{(l-1)}(X,L)$ i.e., i^{th} node output depends upon neighbors distant up to $k\,x\,(l-1)$ hop away. Then in

$$f^{(l)}(X,L) = \delta(g(f^{(l-1)}(X,L),L)W^{(l)}$$

We focus on term,

$$g(X,L) = [f^{(l-1)}(X,L), \dots, L^k f^{(l-1)}(X,L) \quad (6)$$

Particularly the last term involving $L^k f^{(l-1)}(X,L)$. Matrix multiplication of L^k with $f^{(l-1)}(X,L)$ will result in i^{th} node to include all node information which are at most k-hop distance away. But since a node in $f^{(l-1)}(X,L)$ at a distance $k\,x\,(l-1)$ hops, we have i^{th} node containing information at most $k + k(l-1) = kl$ hops distance away.

GCNN model with l layers can capture only kl-hop local-hood structure information at each node. Thus employing GCNN for graph classification with say aggregation layer can capture only average variation of kl-hop local-hood information over the whole graph. To include more global information about the graph one can either increase k (i.e, choose higher order graph convolution filter) or l (i.e, the depth of GCNN model). Both these choice make the model complex and require more data sample to reach satisfying result. However among the two, we prefer increasing the depth of GCNN model because the first choice leads to increase in the breadth of GCNN layer and based on the current understanding of deep learning theory, increasing the depth is favored more over the breadth.

For cases where graph node features are missing, like social network datasets, it is a common practice to take node degree as a node feature. Such practice can work for the problems like graph semi-supervised where local structure information drives node output labels (or classes). But in graph classification global features governs the output labels and hence taking node degree is not sufficient. Of course, we can go for a very deep GCNN model that requires higher sample complexity to achieve satisfying results.

We propose to incorporate FGSD features in our GCNN-CapsNet model computed at each node. FGSD features capture global information about the graph and can also be computed in fast manner. Specifically, at each i^{th} node FGSD features are computed as histogram of

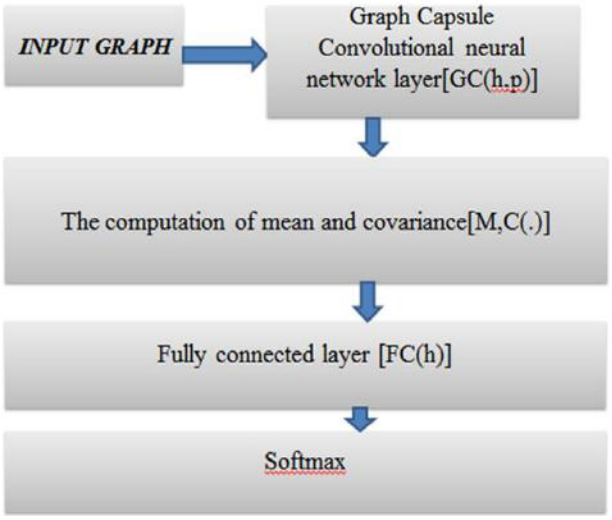
https://dx.doi.org/10.22161/ijaers.6.1.11 *ISSN: 2349-6495(P) | 2456-1908(O)*

the multi-set formed by taking the harmonic distance between all nodes and the i^{th} node. It is given by,

$$S(x,y) = \sum_{n=0}^{N-1} \frac{1}{\lambda n}(\emptyset n(x) - \emptyset n(y))^2 \quad (7)$$

Where $S(x,y)$ is the harmonic distance, x, y are any graph node and $\lambda n, \emptyset n(.)$ is the n^{th} eigenvalue and eigenvector respectively.

IV. GCNN-CAPSNET MODEL CONFIGURATION



Here GC(h,p) is a graph capsule CNN layer with h hidden dimensions and p instantiation parameters. As mentioned earlier, we take the intermediate tensor which is subsequently pass through [M,C(.)] layer which computes means and covariance of the input. Output of [M,C(.)] layer is the passed to two fully connected FC layers with h output dimensions and finally connect to a softmax layer for computing calss probabilities.

4.1 Dataset

Evaluating the GCNN-CapsNet model we perform graph classification task on variety of benchmark datasets. In first round we used bioinformatics datasets namely: PROTEINS, NCI109,NCI1 and ENZYMES. In the second round we used social netrork datasets namely: COLLAB,IMDB-BINARY,REDDIT-BINARY and REDDIT-MULTI5K.

4.2 Experimental setup

All experiments were performed on a single machine loaded with 2xNVIDIA TITAN VOLTA GPUs and 64 GB RAM. And we compare our method with both deep learning models and graph kernels.

Our experiment, we employ these features only for datasets where node feature are missing. Although this strategy can always be used by concatenating FGSD features with original node feature values to capture more global information. Finally our whole end to end GCNN-CapsNet learning model is guaranteed to produce the same output for isomorphic graphs

For deep learning approaches, we adopted 3 recently proposed state-of-art graph convolutional neural network namely: PATCHYSAN (PSCN)[15],Diffusion CNNs(DCNN)[16], Dynamic Edge CNN(ECC)[17].
For graph kernel we adopted 4 state-of-art graphs kernels for comparison namely: Random Walk(RW) [18], Shortest Path Kernel(SP)[19] , Graphlet kernel (GK)[20], Weisfeiler-Lehman Sub-tree Kernels(WL)

4.3 Graph Classification Results

From table 1, it is clear that our GCNN-CapsNet model consistently outperforms most of the considered deep learning methods on bioinformatics datasets with a significant margin of **1% -6%** classification accuracy gain on NCI1 datasets. Again ,this trend is continued to be the same on social network datasets as shown in Table2. Here we were able to achieve up to **4%** accuracy gain on COLLAB dataset and rest were around **1%** gain with consistency when compared against other deep learning approaches.

Our GCNN-CapsNet is also very competitive with state-of-art graph kernel methods. It again show a consistent performance gain of **1% -3%** accuracy on many bioinformatics datasets when compared against with strong graph kernels. While other considered deep learning methods are not even close enough to beat graph kernels on many of these datasets. It is worth mentioning that the most deep learning models are also scalable while graph kernels are more fine-tuned towards handling small graphs.

For social network datasets, we have a significant gain of at least **4% -9%** accuracy(highest being on REDDIT-MULTI dataset) against graph kernels as observed in Table 2. But this is expected as deep learning methods tend to do better with the large amount of data available for training on social network datasets. Altogether, our GCNN-CapsNetmodel shows very promising result against both the current state-of-art deep learning methods and graph kernels.

Dataset	PROTEINS	NCI109	NCI1	ENZYMES
No.Graphs	1113	4127	4110	600
Max.Graph Size	620	111	111	126
Avg.Graph Size	39.80	29.60	29.80	32.60

https://dx.doi.org/10.22161/ijaers.6.1.11 *ISSN: 2349-6495(P) | 2456-1908(O)*

Deep Learning Methods				
PSCN[2016]	75.00±2.51	-	76.34±1.68	-
DCNN[2016]	61.29±1.60	57.47±1.22	56.61±1.04	42.44±1.76
ECC[2017]	-	75.03	76.82	45.67
GCNN-CapsNet	**76.40±4.17**	**81.12±1.28**	**82.72±2.38**	**61.83±5.39**
Graph Kernels				
RW[2003]	74.22±0.42	>73.00±0.21	>1Day	24.16±1.64
SP[2005]	75.07±0.54	73.00±0.21	73.00±0.24	40.10±1.50
GK[2009]	71.67±0.55	62.60±0.19	62.28±0.29	26.61±0.99
WL[2011]	74.68±0.49	**82.46±0.24**	82.19±0.18	52.22±1.26
GCNN-CapsNet	**76.40±4.17**	81.12±1.28	**82.72±2.38**	**61.83±5.39**

Table 1.Classification accuracy on bioinformatics datasets. Result in bold indicates the best reported classification accuracy. Top table compares results with various deep learning approaches while bottom half compares results with graph kernels. '>1day' represent that the computation exceed more than 24hrs

Dataset	**COLLAB**	**IMDB-BINARY**	**REDDIT-BINARY**	**REDDIT-MULTI5K**
No.Graphs	5000	1000	2000	5000
Max.Graph Size	492	136	3783	3783
Avg.Graph Size	74.49	19.77	429.61	508.5

Deep Learning Methods				
PSCN[2016]	72.60±2.15	71.00±2.20	86.30±1.58	49.10±0.70
DCNN[2016]	52.11±0.71	49.06±1.37	OMR	OMR
GCNN-CapsNet	**77.71±2.51**	**71.69±3.40**	**87.61±2.51**	**50.10±1.72**
Graph Kernels				
GK[2009]	72.84±0.28	65.87±0.98	77.34±0.18	41.01±0.17
GCNN-CapsNet	**77.71±2.51**	**71.69±3.40**	**87.61±2.51**	**50.10±1.72**

Table 2.Classification accuracy on social network datasets. Result in **bold**indicates the best reported classification accuracy.Top table compares results with various deep learning approaches while bottom half compares results with graph kernels. '>1day' represent that the computation exceed more than 24hrs. 'OMR' is out of memory error.

V. CONCLUSION

In this paper, we present a novel Graph convolutional neural network with capsule network (GCNN-CapsNet) model based on the fundamental capsule idea to address some of the basic weaknesses of existing GCNN models. Our GCNN-CapsNet model design captures more local structure information than traditional GCNN and can provide much richer representation of individual graph nodes or for the whole graph. For our purpose we employ a capsule function that preserves statistical moment's formation since they are faster to compute.

We propose a novel permutation invariant layer based on computing covariance in our GCNN-CapsNet architecture to deal with graph classification problem which most GCNN models find challenging. This covariance can again be computed in a fast manner and has shown to be better than adopting aggregation or max pooling layer. We also propose to equip our GCNN-CapsNet model with FGSD features explicitly to capture more global information in absence of node features. We finally show GCNN-CapsNet superior performance on many bioinformatics and social network datasets in comparison with existing deep learning methods as well as strong graph kernels and set the current state-of-art.

REFERENCES

[1] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schonauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. Bioinformatics 21, 1 (2005), i47–i56

[2] David K. Duvenaud, DougalMaclaurin, Jorge Aguilera-Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams.

https://dx.doi.org/10.22161/ijaers.6.1.11 *ISSN: 2349-6495(P) | 2456-1908(O)*

2015. Convolutional networks on graphs for learning molecular fingerprints. In Proceedings ofthe Twenty-Eight Annual Conference on Neural Information Processing Systems.2224–2232

[3] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In Proceedings of the Fourth International Conference on Web Search and Web Data Mining. 635–644.

[4] JieBao, Tianfu He, SijieRuan, Yanhua Li, and Yu Zheng. 2017. Planning Bike Lanes based on Sharing-Bikes' Trajectories. In Proceedings of the Twenty-Second ACM SigKDD International Conference on Knowledge Discovery and Data Mining

[5] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schonauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. Bioinformatics 21, 1 (2005), i47–i56

[6] GiannisNikolentzos, PolykarposMeladianos, and MichalisVazirgiannis. 2017. Matching Node Embeddings for Graph Similarity. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. 2429–2435.

[7] Karsten M. Borgwardt and Hans-Peter Kriegel. 2005. Shortest-Path Kernels on Graphs. In Proceedings of the Fifth IEEE International Conference on Data Mining .74–81.

[8] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics. 488–495

[9] RisiKondor and Horace Pan. 2016. The MultiscaleLaplacian Graph Kernel. In Proceedings of the Twenty-Ninth Annual Conference on Neural Information Processing Systems . 2982–2990.

[10] David K. Duvenaud, DougalMaclaurin, Jorge Aguilera-Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan AspuruGuzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In Proceedings of the Twenty-Eight Annual Conference on Neural Information Processing Systems 2224–2232.

[11] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs. In Proceedings of the Thirty-Third International Conference on Machine Learning . 2014–2023.

[12] Pinar Yanardag and S. V. N. Vishwanathan. 2015. Deep Graph Kernels. In Proceedings of the Twenty-First ACM SigKDD International Conference on KnowledgeDiscovery and Data Mining. 1365–1374

[13] Hinton, Geoffrey E, Krizhevsky, Alex, and Wang, Sida D. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

[14] Kondor, Risi and Jebara, Tony. A kernel between sets of vectors. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 361–368,2003.

[15] [Niepert, Mathias, Ahmed, Mohamed, and Kutzkov, Konstantin. Learning convolutional neural networks forgraphs. In *Proceedings of the 33rd annual internationalconference on machine learning. ACM*, 2016

[16] Atwood, James and Towsley, Don. Diffusion-convolutionalneural networks. In *Advances in Neural Information ProcessingSystems*, pp. 1993–2001, 2016

[17] Simonovsky, Martin and Komodakis, Nikos. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proc. CVPR*, 2017

[18] Gartner, Thomas, Flach, Peter, and Wrobel, Stefan. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pp. 129–143. Springer, 2003.

[19] Borgwardt, Karsten M and Kriegel, Hans-Peter. Shortestpath kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pp. 8–pp. IEEE, 2005.

[20] Kondor, Risi, Shervashidze, Nino, and Borgwardt,Karsten M. The graphlet spectrum. In *Proceedings ofthe 26th Annual International Conference on MachineLearning*, pp. 529–536. ACM, 2009.

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