TOPOLOGICAL BASED CLASSIFICATION USING GRAPH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In colored graphs, node classes are often associated with either their neighbors class or with information not incorporated in the graph associated with each node. We here propose that node classes are also associated with topological features of the nodes. We use this association to improve Graph machine learning in general and specifically, Graph Convolutional Networks (GCN).

First, we show that even in the absence of any external information on nodes, a good accuracy can be obtained on the prediction of the node class using either topological features, or using the neighbors class as an input to a GCN. This accuracy is slightly less than the one that can be obtained using content based GCN.

Secondly, we show that explicitly adding the topology as an input to the GCN does not improve the accuracy when combined with external information on nodes. However, adding an additional adjacency matrix with edges between distant nodes with similar topology to the GCN does significantly improve its accuracy, leading to results better than all state of the art methods in multiple datasets.

1 INTRODUCTION AND RELATED WORK

One of the central assumptions in node classification tasks is that neighboring nodes have similar classes (Ji & Han, 2012; Berberidis & Giannakis, 2018; Zhu et al., 2003b; Sindhwani et al., 2005). This has been extensively used in node classification tasks (Belkin & Niyogi, 2004; Zhu et al., 2003a). Such approaches are now often denoted as graph neural networks (i.e. machine learning where the input is a graph/network) (Scarselli et al., 2008; Gori et al., 2005; Li et al., 2015). Four main approaches have been proposed to take advantage of a graph in machine learning:

- Regularize the output requiring that neighboring nodes either in the graph or in its projection have similar classes.
- Use the graph to propagate labels and learn the best propagation.
- Use the graph to project the nodes to real valued vectors and use those for supervised or unsupervised learning.
- Use Graph Convolutional Networks (GCN) for convolutions on the input of a node and its neighbors.

Regularization or graph partitioning methods include among others partitioning the graphs based on the eigenvalues of the Laplacian (assuming that nodes with the same partition have similar classes). The Laplacian of a graph is : $L = D - A$, where D is a diagonal matrix, with the sum of each row in the diagonal and A is the adjacency matrix. This Laplacian is often weighted by multiplying it on the left and the right by D to normalize for the degree (Dhillon et al., 2007; Karypis & Kumar, 1995). Other works have used variants of this idea, each using smoothness and graph distance differently (Belkin & Niyogi, 2004; Sindhwani et al., 2005). An alternative approach is to use quadratic penalty with fixed labels for seed nodes (Zhou et al., 2004; Zhu et al., 2003a).

Multiple diffusion and information propagation models have also been proposed either through explicit diffusion, or through the projection of nodes into real valued vectors (Rosenfeld & Globerson, 2017). For example, DeepWalk (Perozzi et al., 2014), where a truncated random walk is performed on nodes. It then uses these sentences as an input to skipgram to compute a projection of each

word into R^N, maximizing the sentence probability. Planetoid Yang et al. (2016) also uses random walks combined with negative sampling. Duvenaud et al. (2015) uses a translation of subgraphs to hash functions for a similar task in the context of molecule classifications. A very similar approach was presented by Grover & Leskovec (2016) (Node2Vec) by projecting nodes minimizing the distance of neighbored nodes in a truncated random walk. The DNGR model (Cao et al., 2016) uses random walk to compute the mutual information between points (the PPMI-positive pointwise mutual information), and then a SVD decomposition to project into space. PPMI was used for word representations in Levy et al. (2015) and is a sparse high dimensional representation.

Another possible approach is the projection of the graph (often using the Laplacian eigenvectors), and the usage of the projection for classification (and not only for a smoothness based regularization), where either the graph itself is used (in such a case, the eigenvectors themselves are used) or an input to the graph is used. In such a case, a convolution with these eigenvectors was used (Masci et al., 2015; Monti et al., 2017). A Multi-Dimensional-Scaling (MDS) projection of the points in the graphs was also used for a similar goal (Belkin & Niyogi, 2002; Levy et al., 2015). Alternative approaches were inspired again by word embedding methods (Mikolov et al., 2013) such as word2vec. These methods use the graph to define a context in relation to which the node embedding is constructed. When the data includes only the graph, the embeddings are used as features and fed into existing predictors (Perozzi et al., 2014). These methods can be thought of as propagating features rather than labels. Henderson et al. (2011) defines local features to translate each node to a features vector and use those to predict classes.

Recently, Kipfs and collaborators, in a seminal work, proposed a simplification of spectral based convolutions (Kipf & Welling, 2016; Schlichtkrull et al., 2018), and instead use a two-layer approach, which can be summarized as:

$$X_{n+1} = \sigma(\tilde{A} \times X_n \times W_n), \tag{1}$$

where $\tilde{A}$ is a normalized adjacency matrix: $\tilde{A} = D^{-1/2}[A + A^T + I]D^{-1/2}$. They test their work on multiple graphs with labeled nodes including CiteSeer, Cora, Pubmed, and Nell.

Convolution approaches can also be used with the graph as a filter on the input. Most such convolutions are spectral (use the Laplacian eigenvectors). However, recent methods are based on random filters. Those include among others: Atwood & Towsley (2016) which defines predetermined convolutions with powers of the adjacency matrix and then combines these powers using learned weights to maximize the classification precision of either the full graph or the classification of nodes. Bruna et al. (2013) provide a multi-level graph convolution with pooling, where at each stage nodes are merged into clusters using agglomerative clustering methods, and combine it with a pooling method to represent the different resolution of images. This has been extended (Henaff et al., 2015; Bronstein et al., 2017) to different convolutional kernels (mainly spectral, but also diffusion-based kernels) and the classification of images, using ImageNet (see Bronstein et al. (2017) for a detailed review of all convolution methods). Vandergheynst and collaborators mainly use polynomial convolution in the spectral domain. Similar formalisms were used to study not only single snapshots, but also with recurrent networks time series of graphs, mainly again in image analysis (Seo et al., 2018). 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).

GCNs capture dependencies of nodes' features. However, current techniques consider only local neighborhoods. Thus, long-range dependencies can only be captured when these operations are applied repeatedly, propagating signals progressively through the data. To catch long-range dependencies, Kipf & Welling (2016) proposed stacking multiple layers of GCN. While this is possible in theory, it has never been successfully applied. In practice, GCN models work the best with 2-3 layers (Kipf & Welling, 2016; Monti et al., 2017; Veličković et al., 2017; Levie et al., 2018; Fey et al., 2018). Abu-El-Haija et al. (2018) proposed using NGCN train multiple instances of GCNs over different distances regions. While this led to good performance, it is highly inefficient and does not scale to long distances (as the number of models scales linearly with the desired length).

However, long range correlations can be obtained from a different direction. Recently, a correlation has been shown between the topological attributes (e.g. degree, centrality, clustering coefficient...) of nodes and their class (Shi & Malik, 2000; Yang et al., 2013; Cannistraci et al., 2013; Rosen &

Louzoun, 2015; Naaman et al., 2018). Inspired by the improvement of non-local operations in a variety of tasks in the field of computer vision Wang et al. (2018), we propose a novel non-local operation for GCN, based on the topology of the graph. Our operation is generic and can be implemented with every GCN to capture long-range dependencies, allowing information propagation to distant nodes.

There are several advantages of using non-local operations: (a) In contrast to the standard local convolution layer, non-local operations capture long-range dependencies directly by computing interactions between any two nodes, regardless of their positional distance; (b) As we show in experiments, non-local operations are efficient and achieve their best results even with only a few layers; (c) Finally, our non-local convolution can be easily combined with other graph convolution techniques (e.g. GCN, GAT).

2 Main Contributions Of The Current Work

We here propose the following contributions of nodes topology to graph-based machine learning.

First, we show that in the absence of external information, node topology can be used to predict the class of nodes using a feed-forward network. The topology of a node is represented by a vector of attributes of each node, including among others, its degree, the frequency of different sub-graphs around it and its centrality.

We then show that this can be translated to GCN through an input representing the number of first and second neighbors belonging to each class in the training set.

Finally, we show in the context of GCN, that it is better to add an additional adjacency matrix representing the similarity between node topologies to the GCN than actually adding the topology of the nodes as an input.

GCN and Graph Attention Networks (GAT) with this additional adjacency matrix produce accuracies better than all state of the art methods on the Cora, Pubmed, and CiteSeer Datasets.

3 Models And Data

3.1 Datasets studied

Following Shchur et al. (2018), we used four well-known citation network datasets: PubMed, CiteSeer and CORA (Yang et al., 2016), as well as the extended version of CORA from Bojchevski & Günnemann (2017), denoted as CORA-Full, and two co-authorship networks: Coauthor CS, Coauthor Physics. Descriptions of these datasets, as well as statistics, can be found in Appendix A.1.

3.2 Network structure

We used the standard GCN model developed by Kipf & Welling (2016) or the GAT (Veličković et al., 2017).

Each GCN layer is defined as in Eq. 1, where $\tilde{A}$ is defined above, X_n is the input from the previous layer, and W_n are the weights of the current layer.

In GAT, each layer may contain multiple heads. A GAT head is a linear combination of nodes' features followed by a non linear function:

$$h'_i = \sigma(\sum_{j \in N_i} \alpha_{i,j} W h_j) \tag{2}$$

where h_j is a set of features for node j, W is a weight matrix, σ is a non linear function, and $\alpha_{i,j}$ are the normalized attention coefficients. Attention coefficients are calculated for each pair of connected nodes to be:

$$\alpha_{i,j} = a(Wh_i, Wh_j) \tag{3}$$

where a is a single layer feed forward network, and W is weight matrix.

The extensions we propose to the model come from either changing the input of the model or from altering $\tilde{A}$. The following modifications were considered:

- Topology based GCN (T-GCN). We extend graph convolution operation to propagate information through distant neighbors with similar topological features. We construct a dual graph with the same nodes as the original graph, but different edges representing the topological similarity of nodes. Nodes with similar topology are connected with an undirected edge. There are many ways to construct those topological edges. Here we chose to present each node as a R^N vector of topological attributes and connect each node to its k most similar nodes (see Appendix A.3 for the full description of attributes used to define the topology of a node). The T-GCN includes two GCN layers performed simultaneously on the input (external features). One GCN uses the regular adjacency matrix. The second uses the dual graph. These two outputs are then concatenated to serve as an input for the next layer (typically, a standard GCN on the original graph). The network's structure is illustrated at Fig 1.
- Topology based GAT (T-GAT). The same as T-GCN, with GAT layers instead of GCN layers (Eq. 2 instead of Eq.1).

We have also tested the following two alternative methods to use the topology. However, both produce lower accuracies than the standard GCN.

- Asymmetric GCN (A-GCN). We incorporate the direction of directed networks by taking the adjacency matrix (asymmetric in directed graph) and concatenate its transpose to it creating a $2n \times n$ matrix : $\tilde{A} = [A + I]|[A^T + I]$,. The dimension of the output of each layer is: $[(2N \times N) \times (N \times i_n) \times (i_n \times o_n)] = 2N \times o_n$., which in turn is passed to the next layer following a rearrangement of the output by splitting and concatenating it to change dimensions from - $2N \times O_n$ to $N \times 2O_n$. For more details about the parameters see Appendix A.2. Multiple inputs were tested in these configurations as detailed in Appendix A.3.
- Combined GCN (C-GCN): This model includes two input types: a topology features matrix and an external features matrix (in the Cora and Citeseer case, the bag-of-words features). First, we pass the data matrix through a GCN layer, which leads to a $2n \times L_1$ output. The two inputs (topology and external features) are then concatenated following a rearrangement of the processed data matrix by splitting in dimension 0 and concatenating in the dimension $12n \times L_1 \rightarrow n \times 2L_1$. Following the concatenation, an $n \times (2L_1 + T)$ matrix is obtained, which is passed forward to the A-GCN layers. The following layers are as above in the A-GCN. For more details about the parameters see Appendix A.2. Multiple inputs were tested in these configurations as detailed in appendix A.3.

4 Experimental Set-Up

As proposed by Monti et al. (2017) and Veličković et al. (2017), we used one set of hyper-parameters for Cora, and used the hyper-parameters optimized for Pubmed for all other networks: For the Cora dataset we used the following parameters. In the T-GCN, we used 1 hidden layer of size 32 for each graph (original and dual). For the T-GAT we chose 16 internal nodes for the regular and 8 internal nodes for the dual graph. We also chose 8 heads (8 independent attention mechanisms, see Veličković et al. (2017) for more details) for both operations at the first layer, and 1 head for the last layer (same as the original GAT). For the other datasets, we used the optimal parameters found for PubMed: 1 hidden layer with a size of $64 + 16$ for T-GCN, and $16 + 16$ features for T-GAT. We used 16 heads on the original operation, 8 heads at the dual operation at the first layer, and 8 heads at the last layer (same as the original GAT).

The first layer activation function was ReLU for T-GCN, and TanH for T-GAT (except for T-GAT for Cora which we used also ReLU). SoftMax was performed on the last layer of all models. Note that for T-GAT, the external features were normalized and GAT heads were concatenated on the first layer, and averaged on the last layer (same as the original GAT). See a summarized table of all parameters in Appendix A.2.

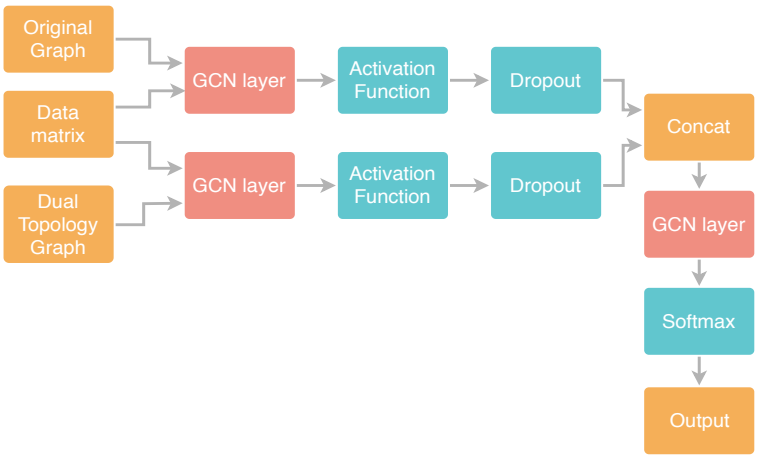


Figure 1: *Schematic depiction of 2-layer Topology based Graph Convolutional Network (T-GCN). The first layer includes 2 parallel layers of GCN: the first on the original graph, and the second on the dual graph. The two outputs are then concatenated and used as an input to the second layer. The second layer is a standard GCN on the original graph*

The input to all classification tasks was a Bag Of Words (BOW) or similar textual description. In the A-GCN and C-GCN, we included a vector of topological features (e.g. degree, clustering coefficient...). An alternative input tested is a vector of the number of first and second neighbors belonging to each class in the training set. For example, assume a classification task with 3 possible labels, and a node with 5 neighbors and 30 second neighbors. Further assume that 1 of the first neighbors belongs to the training set and has label A, 3 of the second neighbors belong to the training set and have label A and 1 of the second neighbors belong to the training set and has label B. The input to the node would be [1,0,0,3,1,0], where the first three values represent the first neighbors and the last three values represent the second neighbors. See Appendix A.3 for more details.

5 Results

5.1 Topology is correlated with class

To test that neighbor class and the node self-topology (as shall be further defined) are correlated with the node class, we performed two tests. We first computed the relative frequency of classes in neighbors, given the class of the node:

$$p(neighbor\ has\ class\ i \mid current\ node\ has\ class\ j) \tag{4}$$

(Fig 2 lower plots). In the absence of correlations, one would expect a flat value, while an absolute correlation would produce an identity matrix. In the Cora or Citeseer networks, the mass of the diagonal is 60 % of the mass (compared with an expected 15 %).

To test for the relation between node topology and class, we computed the average value of multiple topological features (Appendix A.3) for nodes with a given class (in the current context manuscripts belonging to a certain field). Except for the betweenness centrality, the only topological features correlated with class were 3 and 4 small scale motif frequencies. To test for that, a Kruskal Wallis non-parametric test was performed to test for the relation between the node class (manuscript field) and the distribution of features, Over sixty different small scale motifs are associated with the node class (Fig 2 upper plots) .

To test that topology and information propagation can be used to classify node classes, we introduced the topological features above, and the number of neighbors belonging to the training set with a given class as input to a Feed Forward-Network (see Appendix A.4). These two types of information by themselves can be used to classify the class of nodes quite precisely (see Appendix 4).

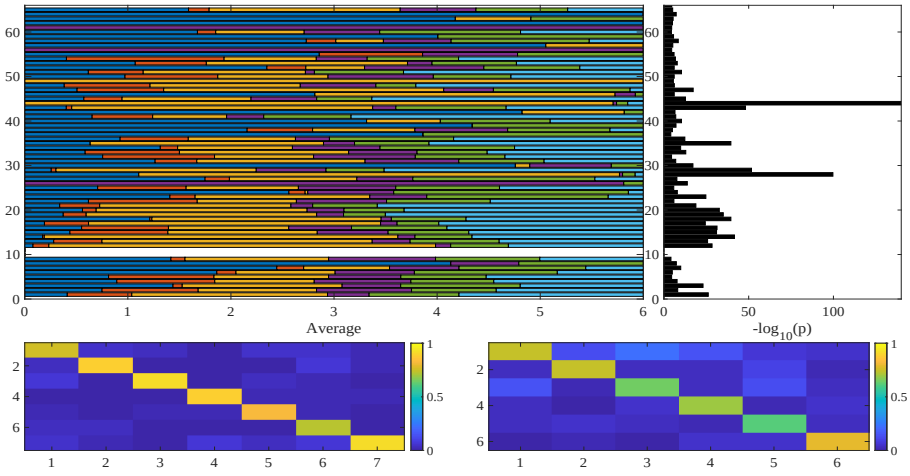


Figure 2: *A) upper right plot - Log p value of non-parametric Kruskal Wallis (KW) test for the association of each topological feature with the class of the manuscripts in CiteSeer. The x axis is the feature number. One can clearly see that some features are highly associated with the manuscript class. Upper left plot Average of each topological feature for nodes belonging to a given class. All values were stacked and normalized to 1. An equal distribution would produce equally divided columns. The upper group are 4 node subgraph frequencies, and the lower group (separated by empty row) are 3 node motifs. Except for the centrality, no other node feature had a a significant KW p value after multiple measurements correction. Lower plots - Correlation of CiteSeer and Cora manuscript class with the neighboring manuscript class. The color in each row represents the fraction of nodes neighboring a given class that belong to all possible classes. One can clearly see the dominating diagonal representing the fact that neighboring nodes tend to have the same color.*

Since the main topological factors correlated with the class are small scale motif, we propose an alternative method to test their contribution to the classification. In order to avoid the explicit computation of sub graph frequencies, which can be computationally expensive, an indirect computation of the topology can be proposed. A simple way to describe such local features is though operations on products of the adjacency matrix. For example, the number of triangles $i \rightarrow j$, $i \rightarrow k$ and $j \rightarrow k$, are the and combination of A and $A * A$ (Fig 4). Thus, instead of explicitly computing such features, one can use as input to the FFN combinations of these products on a one hot representation of the training set class. Formally, let us define for node i, the vector v_i , where v_i^j is the number of neighbors of node i that are in the training set and are of class j, and V is the matrix of all vectors v_i . To this, we add a last constant value to the vector, as shall be explained. We then use different combinations of $A \times V, A^T \times V, A \times A^T \times V$ etc. as inputs to an FFN (see Methods). When these products are applied to the last row (a constant value), they simply count sub-graphs. However, when multiplied by the other component, the sub-graphs composed of a specific class are counted (Fig 4 B). The accuracy obtained for such products outperforms the only explicit topological measures, or information propagation (Fig 4 upper plot).

5.2 Neighbors class is a better predictor than topology in the absence of external information

Given the correlation between the node's topology and class, we tested whether adding the topology by itself or as an additional input to the BOW would increase the prediction accuracy of the node class for the Cora and Citeseer networks. We have tested both symmetric and asymmetric GCN (see description above), and either topological input by itself or combined with the BOW. Within the topological input, we tested three alternatives: the number of first and second neighbors belonging to each class in the training set, the topological features of each node or their combination.

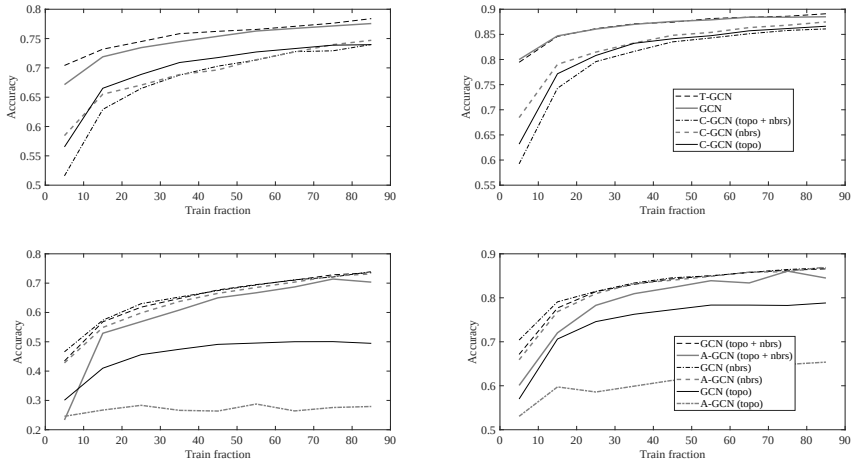


Figure 3: *Average accuracy obtained with 50 random splits as a function of training size (starting from 5%). Validation and test sets were split evenly. Upper plots are for Cora and lower plots for CiteSeer. In the right plots, we compared our asymmetric model with the standard GCN in the absence of external information. Different types of topological features were tested as input (see Results). One can clearly see that the neighbors feature is the best input (performed almost as good as external information), and the standard GCN is better than the A-GCN. In the left plots, we compared the standard GCN with T-GCN and C-GCN (with different types of topological features), where the input is BOW. The C-GCN does not perform well with all three types, and the T-GCN is always equal to or better than the standard GCN.*

As expected, over all tested training set fractions, the models with the BOW outperform the ones without it. Within the models without BOW, ignoring the edge direction, and using only the number of neighbors in each class is better than any other combination. Moreover, combining BOW with topology as input only reduces the accuracy (Fig 3). Still, it is interesting to note that the accuracy without the BOW is not so far from the accuracy with the BOW at high training set fraction (Fig 3). For example, the accuracy for Cora with train size of 55% is 87.9% with BOW, and 85% with only neighbors features as input.

5.3 Model with topology with adjacency matrix outperforms all existing models

Since adding topology as an input did not improve the accuracy, we tested whether using the topology to propagate information between distant nodes based on the similarity of topological attributes helps. We compared our topology-based convolution (T-GCN) to state of the art models on multiple standard real-world networks (see Models and Data). In Cora, CiteSeer, and PubMed networks, we used the previously published split of train-test (Yang et al., 2016), and for Cora-Full and the co-authorship networks we took 20 labeled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the test set (same as Shchur et al. (2018)). We repeated the experiment 100 times and report the average accuracy over all trials. In each trial, we split the data randomly (except for the standard splits where the train is fixed). Parameters for all models can be found in Appendix A.2. All baselines results were copied from the related paper. Furthermore, we report the results of GCN (Kipf & Welling, 2016) and GAT (Veličković et al., 2017) using our own implementation, written using pytorch (which produce slightly lower results than the published result for the same architecture). to fairly evaluate GAT, we used 500 epochs for training. These are the base implementation used for T-GCN and T-GAT. The summary of the results is presented in table 1

One can clearly see that the T-GCN and T-GAT outperform all other models in Cora, Pubmed, and Physics (Table 1). Moreover, the current comparison was performed using the original split in

Table 1: Results - average accuracy over 100 trials. For Cora, CiteSeer, and PubMed we used the standard splits as Yang et al. (2016). For Cora-Full, Physics, and CS we used $20 \times \#Classes$ random nodes as train, and $30 \times \#Classes$ for validation (Shchur et al., 2018). For Cora-Full and Physics we reported the T-GAT results of only 20 and 10 trials accordingly since they were tested on the CPU

Method	**CiteSeer**	**Cora**	**PubMed**	**Physics**	**CS**	**Cora Full**
DCNN (Atwood & Towsley, 2016)	71.1	81.3	-	-	-	-
Planetoid (Yang et al., 2016)	64.7	75.7	77.2	-	-	-
ChebNet (Defferrard et al., 2016)	69.8	81.2	74.4	-	-	-
GCN (Kipf & Welling, 2016)	70.3	81.5	79	92.8	91.1	**62.2**
Sage (Hamilton et al., 2017)	63.5	77.4	77.6	**93**	**91.3**	58.6
MoNet (Monti et al., 2017)	-	81.7	78.8	92.5	90.8	59.8
GAT (Veličković et al., 2017)	**72.5**	83	79	92.5	90.5	51.9
N-Sage (Abu-El-Haija et al., 2018)	71	81.8	79.4	-	-	-
N-GCN (Abu-El-Haija et al., 2018)	72.2	83	79.5	-	-	-
CayleyNet (Levie et al., 2018)	-	81.9	-	-	-	-
GCN (our imp)	67.1	80.4	78.9	92.9	90.7	59.8
GAT (our imp)	70.9	83	78.5	92.2	89.5	62
T-GCN (ours)	71.3	83	**79.8**	**93**	91.2	62.1
T-GAT (ours)	72.2	**83.8**	79.3	92.8	90.1	61.8

CiteSeer, Cora, and Pubmed. We have tested random splits to check the performance of the T-GCN. Indeed, in the random split, the T-GCN always has a higher accuracy than the GCN (Fig 3) even in the CiteSeer dataset, with a difference that can reach up to 3.3%.

6 CONCLUSIONS

Convolution methods to aggregate information from multiple distances are among the leading image classification methods. In images, most of these convolutions are symmetric and sometimes isotropic around each point. However, in contrast with images that are typically overlaid on a 2D lattice, graphs have a complex topology. This topology is highly informative of the properties of nodes and edges (Rosen & Louzoun, 2015; Naaman et al., 2018), and can thus be used to classify their classes. This complex topology can be combined with convolutional networks to improve their accuracy.

In undirected graphs, the topology can often be captured by a distance maintaining projection into R^N, using unsupervised methods, such as the classical MDS (Kruskal, 1964), or supervised methods to minimize the distance between nodes with similar classes in the training set (Cao et al., 2016). In directed graphs, a more complex topology emerges from the asymmetry between incoming and outgoing edges (i.e., the distance between node i and node j differs from the distance between node j and node i), creating a distribution of subgraphs around each node often denoted sub-graph motifs (Milo et al., 2002). Such motifs have been reported to be associated with both single node/edge attributes as well as whole-graph attributes (Milo et al., 2002). We have here shown that in a manuscript assignment task, the topology around each node is indeed associated with the manuscript class.

In order to combine topological information with information propagation, we proposed a novel GCN where the fraction of second neighbors belonging to each class is used as an input, and the class of the node is compared to the softmax output of the node. This method can indeed produce a high classification accuracy, but less than the one obtained using a BOW input. Moreover, explicitly combining the topology as an input with the BOW reduces the accuracy. However, using the topology to add new edges between nodes with similar topological features actually significantly improves performance in most studied datasets. This suggests that the topology is better used to correlate between the class of distant nodes than to be actually used as an input.

The results presented here are a combination of information propagation and topology-based classification. While each of these two elements was previously reported, their combination into a single coherent GCN based classifier provides a novel content independent method to classify nodes. With the current ever-increasing concerns about privacy, new content independent methods for node classification become essential.

REFERENCES

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

Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. *arXiv preprint arXiv:1802.08888*, 2018.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

Vladimir Batagelj and Matjaz Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591, 2002.

Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1-3):209–239, 2004.

Dimitris Berberidis and Georgios B Giannakis. Data-adaptive active sampling for efficient graph-cognizant classification. *IEEE Transactions on Signal Processing*, 66(19):5167–5179, 2018.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. Minimum curvilinearity to enhance topological prediction of protein interactions by network embedding. *Bioinformatics*, 29(13):i199–i209, 2013.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11): 1944–1957, 2007.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1 (1):269–271, 1959.

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.

Martin G Everett and Stephen P Borgatti. The centrality of groups and classes. *The Journal of mathematical sociology*, 23(3):181–201, 1999.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 869–877, 2018.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

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.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 663–671. ACM, 2011.

Royi Itzhack, Yelena Mogilevski, and Yoram Louzoun. An optimal algorithm for counting network motifs. *Physica A: Statistical Mechanics and its Applications*, 381:482–490, 2007.

Ming Ji and Jiawei Han. A variance minimization criterion to active learning on graphs. In *Artificial Intelligence and Statistics*, pp. 556–564, 2012.

George Karypis and Vipin Kumar. Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

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

Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

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.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

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.

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. Technical report, 2015.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

Lev Muchnik, Royi Itzhack, Sorin Solomon, and Yoram Louzoun. Self-emergence of knowledge trees: Extraction of the wikipedia hierarchies. *Physical Review E*, 76(1):016106, 2007.

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.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Yonatan Rosen and Yoram Louzoun. Directionality of real world networks as predicted by path length in directed and undirected graphs. *Physica A: Statistical Mechanics and Its Applications*, 401:118–129, 2014.

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

Nir Rosenfeld and Amir Globerson. Semi-supervised learning with competitive infection models. *arXiv preprint arXiv:1703.06426*, 2017.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pp. 593–607. Springer, 2018.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pp. 362–373. Springer, 2018.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, pp. 107, 2000.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 824–831. ACM, 2005.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pp. 1151–1156. IEEE, 2013.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003a.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3, 2003b.

A Appendix

A.1 Datasets studied

The citation networks contain scientific papers divided into classes by their research field. Edges describe citations in the data set. BOW is also available to describe each publication in the dataset. BOW can be either a 1/0 vector or a TF/IDF weighted word vector for PubMed.

Coauthor CS and Coauthor Physics are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge 3. Here, nodes are authors, that are connected by an edge if they co-authored a paper, node features represent paper keywords for each authors papers, and class labels indicate the most active fields of study for each author.

Table 2: Datasets statistics

Model	***Nodes***	***Edges***	***Classes***	***Features***
CORA	2,708	5,429	7	1433
CITESEER	3,312	4,732	6	3703
PubMed	19,717	44,324	3	500
Cora-Full	18,703	62,421	67	8710
Co-Author CS	18,333	81,894	15	6805
Co-Author Physics	34,493	247,962	5	8415

A.2 Models Parameters

Here are the parameters used for each of the models. For T-GCN and T-GAT the parameters were optimized for PubMed (as observed by Monti et al. (2017) and Veličković et al. (2017)) except for Cora data for set which we used slightly different parameters (denotes as T-GCN Cora, and T-GAT Cora). The parameters are summarized in Table 3.

In all models, the activation function of the last layer is Softmax. The activation function of the first layer is presented In Table 3. Hidden size X+Y means size of X for the original GCN operator and Y for the GCN on the dual graph. The two outputs are concatenated to a total of X+Y size. GAT heads X,Y,Z means X heads for the original GAT operator, and Y heads for the GAT on the dual graph. Z is the number of heads in the last layer. See Models And Data for more details.

A.3 Networks measures

Our goal is to use the graph structure to classify node colors. Hence, we compute features that are only based on the graph structure, ignoring any external content associated with each node. Those features are used to convert nodes into the appropriate network attribute vector (NAV) (Naaman et al., 2018). Following is a list of attributes used. Note that other attributes may have been used with probably similar results.

Table 3: Models Parameters

Model	T-GCN	T-GCN (CORA)	T-GAT	T-GAT (CORA)
Activation	TanH	ReLU	ReLU	ReLU
Drop Out	0.7	0.6	0.6	0.7
Hidden Size	64+16	32+32	16+16	16+8
Learning Rate	0.01	0.001	0.01	0.01
Weight Decay	0.0005	0.01	0.0005	0.001
Epochs	400	300	400	500
K-NN	8	8	15	10
Normalized Features	True	False	True	True
GAT Heads	-	-	16,8,8	8,8,1

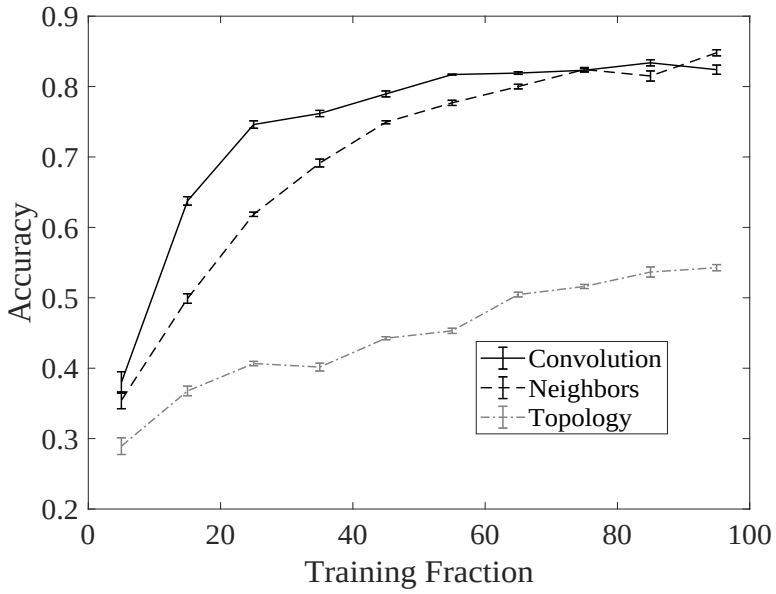
Model	A-GCN	C-GCN
Activation	ReLU	ReLU
Drop Out	0.6	0.6
Hidden Size	100,35	16
Learning Rate	0.01	0.01
Weight Decay	0.001	0.001
Epochs	200	200

- Degree -number of in and out (in case of directed graphs) edges.
- Betweenness Centrality Everett & Borgatti (1999). Betweenness is a centrality measure of a vertex. It is defined by the numbers of shortest paths from all vertices that pass through the vertex.
- Closeness Centrality. Closeness is a centrality measure of a vertex. It is defined as the average length of the shortest path between the vertex and all other vertices in the graph.
- Distance distribution. We compute the distribution of distances from each node to all other nodes using a Djekstra algorithm Dijkstra (1959), and then use the first and second moments of this distribution.
- Flow (Rosen & Louzoun, 2014). We define the flow measure of a node as the ratio between the undirected and directed distances between the node and all other nodes.
- Attraction (Muchnik et al., 2007) . Attraction Basin hierarchy is the comparison between the weighted fraction of the network that can be reached from each vertex with the weighted fraction of the network from which the vertex can be reached.
- Motifs Network motifs are small connected sub-graphs. We use an extension of the Itzhack et al. (2007) algorithm to calculate motifs. For each node, we compute the frequency of each motif where this node participates.
- K-cores (Batagelj & Zaversnik, 2003). A K-core is a maximal subgraph that contains vertices of degree k or more. Equivalently, it is the subgraph of G formed by repeatedly deleting all nodes of degree less than k.
- Louvain community detection algorithm Blondel et al. (2008). The Louvain algorithm is a community detection algorithm. The algorithm works by optimization of modularity, which has a scale value between -1 to 1.

Neighbors Feature. We also used a feature of the training set labels. We summed for each node the number of neighbors belonging to each class in the training set. The sum was represented as a vector of sums (e.g. if a node has 10 neighbors, only three of which are in the training set, with two belonging to the first class, and one belonging to the third class, the vector would be $[2, 0, 1, ..]$). The sum was performed on first and second neighbors producing a vector of twice the number of classes. In a directed graph we calculated two features, one for In neighbors and the second for Out neighbors.

A.4 FEED FORWARD NETWORK

The results in Figure 2 are produced through a feed-forward network with two internal layers of sizes 300 and 100 internal nodes and an output layer with the number of possible classifications (7 and 6 in CiteSeer and Cora, respectively). The nonlinearities were Relus in the internal layers and a linear function in the output layer. An L2 regularization of 0.2 was used for all layers and a 10% drop in our rate. The loss function was a categorical cross-entropy as implemented in Keras with a TensorFlow backend.



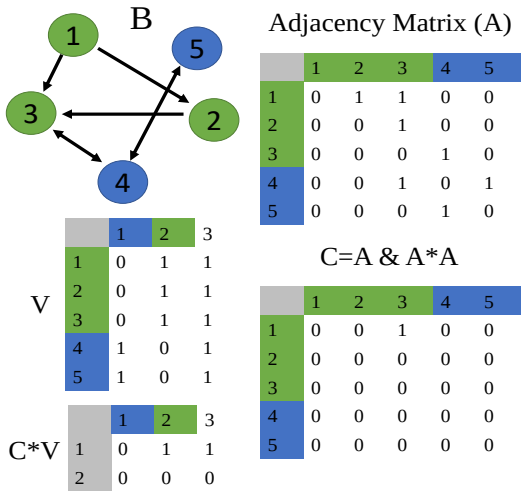


Figure 4: *Upper plot – Accuracy of the CiteSeer classification when only topological features are used (dotted dashed line), when each node is classified using the distribution of classes in its neighbors (dashed), or when products of the neighbors and different combinations of the adjacency matrix are used (full line). Lower plot. Example of sub-graph frequency through adjacency matrix products. Given the graph plotted on the left, the appropriate adjacency matrix (A) and a division of the nodes into green and blue nodes, one can count the number of feed-forward motifs ($x \rightarrow y$ and*