Hierarchically Attentive Graph Pooling with Subgraph Attention

Sambaran Bandyopadhyay [1 2] **Manasvi Aggarwal** [2] **M. Narasimha Murty** [2]

Abstract

Graph neural networks got significant attention for graph representation and classification in machine learning community. Different types of neighborhood aggregation and pooling strategies have been proposed in the literature. In this work, we introduce a higher order hierarchical GNN algorithm (SubGattPool) by employing (i) an attention mechanism which learns the importance and aggregates neighboring subgraphs of a node instead of first-order neighbors, and (ii) a hierarchical pooling strategy which learns the importance of different hierarchies in a GNN. SubGattPool is able to achieve state-of-the-art graph classification performance on multiple real-world datasets.

1. Introduction

Graph neural networks (GNNs) gained significant interest from the research community in the past few years (Defferrard et al., 2016; Kipf & Welling, 2017). A GNN typically uses a message passing framework to update a node representation by aggregating information from its neighbors (Gilmer et al., 2017; Hamilton et al., 2017). For a graph level task such as graph classification (Xu et al., 2019; Duvenaud et al., 2015), GNNs jointly derive the node embeddings and use different pooling mechanisms (Ying et al., 2018; Lee et al., 2019) to obtain a representation of the entire graph.

Attention mechanisms on graphs show promising results for both node classification (Veličković et al., 2018) and graph classification (Lee et al., 2019; 2018) tasks. Typically, attention for neighborhood aggregation in a GNN is computed between a pair of nodes in the immediate neighborhood to capture the importance between them (Veličković et al., 2018; Lee et al., 2018). But in real world situation, calculating importance up to a pair of nodes is not adequate. In molecular biology or in social networks, the presence of particular sub-structures, potentially of varying sizes, in a graph often determines its label. Hence, all the nodes collectively in such a substructure are important, and they may not be important individually or in pairs to classify the graph. In Figure 1, each node (indexed from a to g) in the small synthetic graph can be considered as an agent whose attributes determine its opinion (1:positive, 0: neutral, -1: negative) about 4 products. Suppose the graph can be labelled +1 only if there is a subset of connected (by edges) agents who jointly have positive opinion about all the products. In this case, the blue shaded connected subgraph (a, b, c) is important to determine the label of the graph. Please note that attention over the pairs (Veličković et al., 2018) is not enough as (a, b) cannot make the label of the graph +1 by itself. Also, multiple layers of graph convolution (Kipf & Welling, 2017) with pair-wise attention may not work as the aggregated features of a node get corrupted after the feature aggregation by the first few convolution layers. Besides, recent literature also shows that higher order GNNs that directly aggregate features from higher order neighborhood of a node are theoretically more powerful than 1st order GNNs (Morris et al., 2019). To the best of our knowledge, (Yang et al., 2019) is the only work to propose an attention mechanism on the shortest paths starting from a node to generate the node embedding. However, their computation of shortest path depends on the pairwise node attention and this may fail in the cases when a collection of nodes together is important, but not the individual pairs. With these motivations, we develop a novel higher order attention mechanism which operates in the subgraph level in the vicinity of a node in a GNN framework.

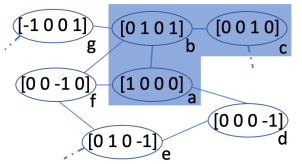


Figure 1. Example to motivate subgraph attention

On the other hand, different types of graph pooling mechanisms (Duvenaud et al., 2015; Morris et al., 2019) have been proposed in the recent GNN literature. Among them, hierarchical graph pooling (Ying et al., 2018) gains significant interest as it is able to capture the intrinsic hierarchical property of several real-world graphs. For example, in a social network, one must model both the ego-networks around individual nodes, as well as the coarse-grained relationships between entire communities (Newman, 2003; Morris et al.,

[1]IBM Research [2]Indian Institute of Science, Bangalore. Correspondence to: S. Bandyopadhyay <samb.bandyo@gmail.com>.

2019). But hierarchical representation (Ying et al., 2018) often fails to perform well in practice mainly because (i) of significant loss of information in learning the sequential hierarchies of a graph when the data is limited; (ii) it treats all the nodes within a hierarchy, and all the hierarchies equally while computing the entire graph representation; (iii) for real-world graphs, the hierarchical structure if often noisy. To address these issues, we again use attention to differentiate different units of a hierarchical graph representation.

Contributions: We propose SubGattPool which (i) employs an attention mechanism to learn the importance and aggregates neighboring subgraphs of a node instead of first-order neighbors, and (ii) a hierarchical pooling strategy which learns the importance of different hierarchies in a GNN. Experimentally, we are able to achieve state-of-the-art graph classification performance on multiple datasets.

2. Proposed Approach: SubGattPool

Figure 2 shows the high-level architecture of SubGattPool. One major component of SubGattPool is the generation of node representations through SubGraph attention (referred as *SubGatt*) layer, which is described below.

2.1. Subgraph Attention Mechanism

The input to the subgraph attention network is an attributed graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ is the set of N nodes and $x_i \in \mathbb{R}^D$ is the attribute vector of the node $v_i \in V$. The output of the model is a set of node features (or embeddings) $h_i \in \mathbb{R}^K$, $\forall i \in [N]$ (K is potentially different from D). We use $[N]$ to denote the set $\{1, 2, \cdots, N\}$ for any positive integer N. We define the immediate (or first order) neighborhood of a node v_i as $\mathcal{N}_i = \{v_j | (v_i, v_j) \in E\}$. For the simplicity of notations, we assume an input graph G to be undirected for the rest of the paper, but extending it for directed graphs is straightforward.

2.1.1. Subgraph selection and Sampling

For each node in the graph, we aim to find the importance of the nearby subgraphs to that node. In general, subgraphs can be of any shape or size. Motivated by the prior works on graph kernels (Shervashidze et al., 2011), we choose to consider only a set of rooted subtrees as the set of candidate subgraphs. So for a node v_i, any tree of the form (v_i), or (v_i, v_j) where $(v_i, v_j) \in E$, or (v_i, v_j, v_k) where $(v_i, v_j) \in E$ and $(v_j, v_k) \in E$, and so on will form the set of candidate subgraphs of v_i. We restrict the maximum size (i.e., number of nodes) of a subtree to T. Also note that, node v_i is always a part of any candidate subgraph for the node v_i according to our design. For example, all possible subgraphs of maximum size 3 for node a in Figure 1 are: (a), (a,b), (a,d), (a,f), (a,b,c), (a,b,f), (a,b,g), (a,d,e), (a,f,e) and (a,f,b).

The number of candidate subgraphs for a node can be very large. For example, the number of rooted subgraphs for the node v_i is $d_{v_i} \times \sum_{v_j \in \mathcal{N}(v_i)} (d_{v_j} - 1) \times \sum_{v_k \in \mathcal{N}(v_j) \setminus \{v_i\}} |\mathcal{N}(v_k) \setminus \{v_i, v_j\}|$, where d_v is the degree of node v and $T = 4$. Clearly, computing attention over these many subgraphs for each node is computationally difficult. So we employ a random subgraph subsampling technique, inspired by the node subsampling techniques for network embedding (Hamilton et al., 2017). More precisely, we randomly select L subgraphs (i.e., subtrees of maximum size T) for each node without replacement if number of candidate subgraphs is more than L, otherwise use round robin sampling. For each node, new sample of subgraphs is taken in each epoch. In any epoch, let us use the notation $\mathbf{S}_i = \{S_{i1}, \cdots, S_{iL}\}$ to denote the set (more precisely it is a multiset as subgraphs can repeat) of sampled subgraphs for node v_i.

2.1.2. Subgraph Attention Network

This subsection describes the attention mechanism for neighborhood aggregation. As mentioned, the node of interest is always positioned as the root of each subgraph generated for that node. To generate a feature for the subgraph, we concatenate the attributes of all the nodes in the subgraph in order. If the subgraph has less than T nodes, we append zeros at the end to ensure equal length feature vector for all the subgraphs. For example, if the maximum size of a subgraph is $T = 4$, then the feature of the subgraph (v_i, v_j, v_k) is $[x_i||x_j||x_k||0] \in \mathbb{R}^{4D}$, where $||$ is the concatenation operation and 0 is the zero vector in $\mathbb{R}^D$. Let us denote this derived feature vector of any subgraph S_{il} as $\hat{x}_{il} \in \mathbb{R}^{TD}$, $\forall i \in [N]$ and $\forall l \in [L]$. Next, we use self-attention on the features for the sampled subgraphs for each node. As a first step, we use a shared linear transformation, parameterized by a trainable weight matrix $W \in \mathbb{R}^{K \times TD}$, to the feature of all the sampled subgraphs S_{il}, $\forall i \in [N]$ and $\forall l \in [L]$ selected in an epoch. Next we introduce a trainable self attention vector $a \in \mathbb{R}^K$ to compute the attention coefficient α_{il} which captures the importance of the subgraph S_{il} of node v_i, as follows:

$$\alpha_{il} = \frac{exp(\sigma(a^T W \hat{x}_{il}))}{\sum_{l' \in [L]} exp(\sigma(a^T W \hat{x}_{il'}))} \quad , \quad h_i = \sigma\Big(\sum_{l=1}^{L} \alpha_{il} W \hat{x}_{il}\Big) \tag{1}$$

We have used Leaky ReLU as the activation function $\sigma()$ for all the experiments. α_{il} gives normalized attention scores over the set of sampled subgraphs for each node. We use them to compute the representation h_i of node v_i as shown in Eq. 1. Needless to say, one can easily extend the above subgraph attention by multi-head attention by employing few independent attention mechanisms of Eq. 1 and concatenate the resulting representations (Vaswani et al., 2017).

This completes one full subgraph attention layer. We can stack such multiple layers to design a full SubGatt network.

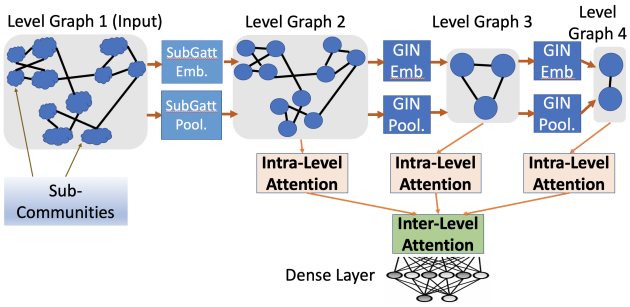


Figure 2. SubGattPool Network for graph classification

2.2. Hierarchically Attentive Graph Pooling

The hierarchical structure of SubGattPool is motivated by (Ying et al., 2018). As shown in Figure 2, there are $R = 4$ different levels of the graph in the hierarchical architecture. The first level is the input graph. Let us denote these level graphs by $G^1, \cdots, G^R$. There is a GNN layer between the level graphs G^r and G^{r+1} which comprises of an embedding layer to generate the embedding of the nodes of G^r and a pooling layer which maps the nodes of G^r to the nodes of G^{r+1}. Please note, number of nodes N_1 in the first level graph depends on the input graph, but we keep the number of nodes N_r in the consequent level graphs G^r ($\forall r = 2, \cdots, R$) fixed for all the input graphs (in a graph classification dataset), which help us to design the shared hierarchical attention mechanisms, as discussed later. As pooling mechanisms shrink a graph, $N_r > N_{r+1}, \forall r \leq R-1$.

Any level graph G^r is defined by its adjacency matrix $A_r \in \mathbb{R}^{N_r \times N_r}$ and the feature matrix $X_r \in \mathbb{R}^{N_r \times K}$. The rth embedding layer and the pooling layer are defined by:

$$Z_r = \begin{cases} \text{SubGatt}_{embed}(A_r, X_r) \ , & r = 1 \\ \text{GIN}_{r,embed}(A_r, X_r) \ , & r > 1 \end{cases}$$

$$P_r = \begin{cases} \text{softmax}(\text{SubGatt}_{pool}(A_r, X_r)) \ , & r = 1 \\ \text{softmax}(\text{GIN}_{r,pool}(A_r, X_r)) \ , & 1 < r \leq R-1 \end{cases} \tag{2}$$

Here, $Z_r \in \mathbb{R}^{N_r \times K}$ is the embedding matrix of the nodes of G^r. The softmax after the pooling is applied row-wise. (i,j)th element of $P_r \in \mathbb{R}^{N_r \times N_{r+1}}$ gives the probability of assigning node v_i^r in G^r to node v_j^{r+1} in G^{r+1}. Based on these, graph G^{r+1} is constructed as, $A_{r+1} = P_r^T A_r P_r$ and $X_{r+1} = P_r^T Z_r$.

As the embedding and pooling GNNs, we use SubGatt networks (Section 2.1) only after the level graph 1. This is because other level graphs G^r ($r > 1$) have more number of soft edges (i.e., with probabilistic edge weights) due to use of softmax at the end of pooling layers. Hence, the number of neighboring rooted subtrees will be high in those level graphs and the chance of having discrete patterns will be less. We use GINs (Xu et al., 2019) as the embedding and pooling GNNs for G^r, $r > 1$. lth layer of GIN can be defined as: $h_v^{l+1} = MLP^l\Big((1+\epsilon^l)h_v^l + \sum_{u \in \mathcal{N}(v)} h_u^k\Big)$. Here, $h_v^{l+1} \in \mathbb{R}^K$ is the hidden representation of the node v in $l+1$th layer of GIN and ϵ is a learnable parameter.

Intra-level attention layer: To alleviate the problem of information loss and noisy hierarchical nature of graphs, we propose to use attention mechanisms in the hierarchical graph pooling, to combine features from different level graphs in the architecture. We consider level graphs G^2 to G^R for this, as their respective numbers of nodes are same across all the graphs in a dataset. We introduce *intra-level attention layer* to obtain a global feature for each level graph $G^r, \forall r = 2, \cdots, R$. More precisely, we use the convolution based self attention within the level graph G^r as:

$$e_r = \text{softmax}(\widetilde{D}_r^{-\frac{1}{2}} \widetilde{A_r} \widetilde{D}_r^{-\frac{1}{2}} X_r \theta) \ \in \mathbb{R}^{N_r}; \quad x^r = X_r^T e_r \ \in \mathbb{R}^K \tag{3}$$

Here, the softmax to compute e_r is taken so that a component of e_r becomes the normalized (i.e., probabilistic) importance of the corresponding node in G^r. $\widetilde{A_r} = A_r + I_{N_r}$ is the adjacency matrix with added self loops of G^r. $\widetilde{D}$ is the diagonal matrix of dimension $N_r \times N_r$ with $\widetilde{D}(i,i) = \sum_{j=1}^{N_r} \widetilde{A}_{ij}$. $\theta \in \mathbb{R}^K$ is the trainable vector of parameters of intra-level attention, which is shared across all the level graphs $G^r, \forall r = 2, \cdots, R$. Intuitively, θ contains the importance of individual attributes and the components of N_r dimensional $X_r\theta$ gives the same for each node. Finally, multiplying that with $\widetilde{D}_r^{-\frac{1}{2}} \widetilde{A_r} \widetilde{D}_r^{-\frac{1}{2}}$ produces the (normalized) importance of a node based on its own features and the features of immediate neighbors (for one layer of intra-level attention). Hence, x^r is a learnable weighted sum of the features of the nodes in G^r. Representing level graphs separately by the proposed intra-level attention makes their impact more prominent.

Inter-level attention layer: This layer aims to get the final representation, referred as $x^G \in \mathbb{R}^K$, of the input graph from $x_2, \cdots, x_R$; as obtained from the intra-level attention layers. As different level graphs of the hierarchical representation have different importance to determine the label of the input graph, we propose to use the following self-attention mechanism.

$$\widetilde{e} = \text{softmax}(X_{inter}\widetilde{\theta}) \in \mathbb{R}^{R-1} \ \text{ and } \ x^G = X_{inter}^T \widetilde{e} \in \mathbb{R}^K \tag{4}$$

X_{inter} is the $R-1 \times K$ dimensional matrix whose rows correspond to x^r (the output of intra-level attention layer for G^r), $r = 2, \cdots, R$. $\widetilde{\theta} \in \mathbb{R}^K$ is a trainable self attention

Algorithms	MUTAG	PTC	PROTEINS	NCI1	NCI109	IMDB-B	IMDB-M
GK (Shervashidze et al., 2009)	81.39±1.7	55.65±0.5	71.39±0.3	62.49±0.3	62.35±0.3	NA	NA
RW (Vishwanathan et al., 2010)	79.17±2.1	55.91±0.3	59.57±0.1	NA	NA	NA	NA
PK (Neumann et al., 2016)	76±2.7	59.5±2.4	73.68±0.7	82.54±0.5	NA	NA	NA
WL (Shervashidze et al., 2011)	84.11±1.9	57.97±2.5	74.68±0.5	**84.46±0.5**	**85.12±0.3**	NA	NA
AWE-DD (Ivanov & Burnaev, 2018)	NA	NA	NA	NA	NA	74.45±5.8	51.54±3.6
AWE-FB (Ivanov & Burnaev, 2018)	87.87±9.7	NA	NA	NA	NA	73.13±3.2	51.58±4.6
node2vec (Grover & Leskovec, 2016)	72.63±10.20	58.85±8.00	57.49±3.57	54.89±1.61	52.68±1.56	NA	NA
sub2vec (Adhikari et al., 2017)	61.05±15.79	59.99±6.38	53.03±5.55	52.84±1.47	50.67±1.50	55.26±1.54	36.67±0.83
graph2vec (Narayanan et al., 2017)	83.15±9.25	60.17±6.86	73.30±2.05	73.22±1.81	74.26±1.47	71.1±0.54	50.44±0.87
InfoGraph (Sun et al., 2020)	89.01±1.13	61.65±1.43	NA	NA	NA	73.03±0.87	49.69±0.53
DGCNN (Zhang et al., 2018)	85.83±1.7	58.59±2.5	75.54±0.9	74.44±0.5	NA	70.03±0.9	47.83±0.9
PSCN (Niepert et al., 2016)	88.95±4.4	62.29±5.7	75±2.5	76.34±1.7	NA	71±2.3	45.23±2.8
DCNN (Atwood & Towsley, 2016)	NA	NA	61.29±1.6	56.61±1.0	NA	49.06±1.4	33.49±1.4
ECC (Simonovsky & Komodakis, 2017)	76.11	NA	NA	76.82	75.03	NA	NA
DGK (Yanardag & Vishwanathan, 2015)	87.44±2.7	60.08±2.6	75.68±0.5	80.31±0.5	80.32±0.3	66.96±0.6	44.55±0.5
DIFFPOOL (Ying et al., 2018)	83.56	NA	76.25	NA	NA	NA	47.91
IGN (Maron et al., 2018)	83.89±12.95	58.53±6.86	76.58±5.49	74.33±2.71	72.82±1.45	72.0±5.54	48.73±3.41
GIN (Xu et al., 2019)	89.4±5.6	64.6±7.0	76.2±2.8	82.7±1.7	NA	75.1±5.1	52.3±2.8
1-2-3GNN (Morris et al., 2019)	86.1±	60.9±	75.5±	76.2±	NA	74.2±	49.5±
3WL-GNN (Maron et al., 2019)	90.55±8.7	66.17±6.54	**77.2±4.73**	83.19±1.11	81.84±1.85	72.6±4.9	50±3.15
SubGattPool	**93.29±4.78**	**67.13±6.45**	76.92±3.44	82.59±1.42	80.95±1.76	**76.49±2.94**	**52.46±3.48**
Rank	1	1	2	3	3	1	1

Table 1. Classification accuracy (%) of different algorithms (21 in total) for graph classification. NA denotes the case when the result of a baseline algorithm could not be found on that particular dataset from the existing literature. The last row 'Rank' is the rank (1 being the highest position) of our proposed algorithm SubGattPool among all the algorithms present in the table.

vector. Similar to Eq. 3, softmax is taken to convert $\widetilde{e}$ to a probability distribution of importance of different graph levels. Finally, x^G is fed to a classification layer of the GNN, which is a dense layer followed by a softmax to classify the entire input graph in an end-to-end fashion. This completes the construction of SubGattPool architecture.

It can be shown that total number of parameters of SubGattPool network is $O(KTD + RKD)$, which is independent of both the average number of nodes and the number of graphs in the dataset. We use ADAM (with learning rate set to 0.001) on the cross-entropy loss of graph classification to train these parameters. In contrast to existing hierarchical GNNs (Ying et al., 2018; Morris et al., 2019), SubGattPool does not only rely on the last level of the hierarchy to obtain the final graph representation. SubGattPool may have more than 1 node in the last level graph. SubGattPool is also less prone to information loss in the hierarchy and is able to learn importance of individual nodes in a hierarchy (i.e., level graph) and the importance of different hierarchies. In terms of design, we propose subgraph attention mechanism through SubGatt network and use it along with GIN for different embedding and pooling layers of SubGattPool.

3. Experiments on Graph Classification

We use 5 bioinformatics graph datasets (MUTAG - NCI09) and 2 social network datasets (IMDB-BINARY and IMDB-MULTI) to evaluate the performance for graph classification. The details of these datasets can be found in Appendix. To compare the performance of SubGattPool, we choose twenty state-of-the-art baseline algorithms from the domains of graph kernels, unsupervised graph representation and graph neural networks (Table 1). We adopt the same experimental setup as there in (Xu et al., 2019; Maron et al., 2019). This helps us to collect the reported accuracy numbers of the baselines directly from the literature to avoid any degradation of performance due to insufficient parameter tuning.

We keep the values of the hyperparameters to be the same across all the datasets, based on the averaged validation accuracy. We set the pooling ratio (defined as $\gamma = \frac{N_{r+1}}{N_r}$, $\forall r < R-1$) at 0.5, the number of levels R=3 and the maximum subgraph size (T) to be 3. We sample L=12 subgraphs for each node in each epoch of SubGatt. Following most of the literature, we set the embedding dimension K to be 128. We use L2 normalization and dropout in SubGattPool architecture to make the training stable. Table 1 shows that SubGattPool is able to improve the state-of-the-art on MUTAG, PTC, IMDB-B and IMDB-M for graph classification. On PROTEINS, the performance gap with the best performing baseline (which is 3WL-GNN (Maron et al., 2019)) is less than 1%. But on NCI1 and NCI109, WL kernel turns out to be the best performing algorithm with a good margin ($> 1\%$) over all the GNN based algorithms. In terms of standard deviation, SubGattPool is highly competitive and is often better than most of the better performing GNNs.

4. Conclusion

We have proposed a novel GNN based graph classification algorithm called SubGattPool which uses higher order attention over the subgraphs of a graph and also addresses some shortcomings of the existing hierarchical graph representation techniques. We believe this work encourages further development in the area of hierarchical graph representation.

References

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Distributed representations of subgraphs. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 111–117. IEEE, 2017.

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

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

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

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR, 2017.

Grover, A. and Leskovec, J. 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.

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

Ivanov, S. and Burnaev, E. Anonymous walk embeddings. *arXiv preprint arXiv:1805.11921*, 2018.

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

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *International Conference on Machine Learning*, pp. 3734–3743, 2019.

Lee, J. B., Rossi, R., and Kong, X. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674. ACM, 2018.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 2153–2164. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8488-provably-powerful-graph-networks.pdf.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

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

Newman, M. E. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

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

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

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lfF2NYvH.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9: 2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.

Yang, Y., Wang, X., Song, M., Yuan, J., and Tao, D. Spagan: shortest path graph attention network. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4099–4105. AAAI Press, 2019.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

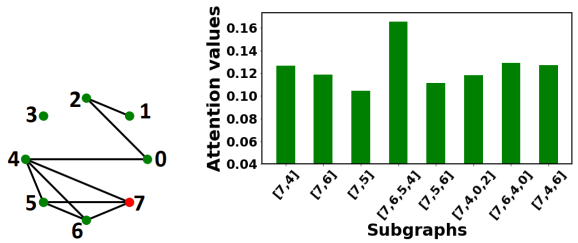
Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A. Insights from a Synthetic Experiment

Subgraph attention is a key component of SubGattPool. Here, we validate the learned attention values on different subgraphs by conducting an experiment on a small synthetic dataset containing 50 graphs, and each graph having 8 nodes. Each graph has 2 balanced communities and exactly for 50% of the graphs, one community consists of a clique of size 4. We label a graph with +1 if the clique of size 4 is present, otherwise the label is -1. We run SubGattPool on this synthetic dataset, with $K = 16$, $T = 4$, $L = 12$ #SubGatt layers=1, $\gamma = 0.75$ and $R = 3$. Once the training is complete, we randomly select a graph and a node in it and plot the attention values of all the subgraphs selected in the last epoch for that node, in Figure 3. Clearly, the attention value corresponding to the clique is much higher than that to the other subgraphs. Hence, our algorithm is able to identify and pay more importance to the structure which determines the label of the graph.

Dataset	#Graphs	#Max Nodes	#Labels	#Attributes
MUTAG	188	28	2	NA
PTC	344	64	2	NA
PROTEINS	1113	620	2	1
NCI1	4110	111	2	NA
NCI109	4127	111	2	NA
IMDB-BINARY	1000	136	2	NA
IMDB-MULTI	1500	89	3	NA

Table 2. Summary of the datasets used in our experiments, further details can be found at (https://bit.ly/39T079X)



(a) Sample Graph (b) Normalized Attention Values

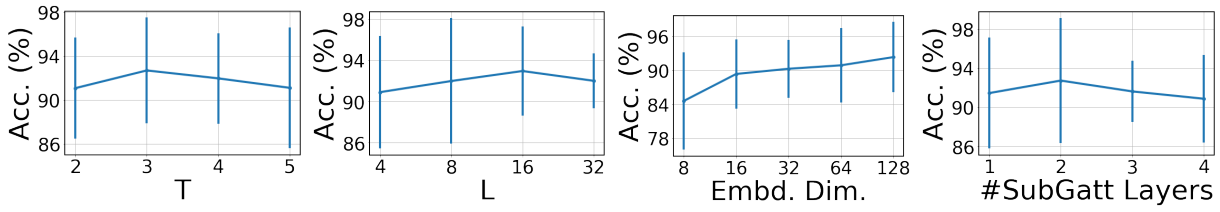
Figure 3. Attention values of different subgraphs selected for the node 7 of the Graph in (a). Clearly, attention to the clique of size 4 is more than all the other subgraphs.

B. Model Ablation Study and Sensitivity Analysis

SubGattPool has mainly two novel components. They are the SubGatt layer, and the intra-level and inter-level attention layers.Figure 4 shows the graph visualization using t-SNE (van der Maaten & Hinton, 2008) on MUTAG. We choose DIFFPOOL as the base model because it is also a hierarchical graph representation technique. Fig. 4(b) and 4(c) explain the incremental improvement by only intra and inter level attentions, and SubGatt layer respectively. Finally, Fig. 4(d) shows the performance by SubGattPool, which combines all these components into a single network. We also conduct sensitivity analysis of SubGattPool w.r.t. all the important hyperparameters, as explained in Figure 5. It turns out that SubGattPool is reasonably stable with respect to them.

(a) DIFFPOOL (b) SubGattPool \SubGatt (c) SubGattPool \I-I-L-A (d) SubGattPool

Figure 4. t-SNE visualization of the graphs from MUTAG (different colors show different labels of the graphs) by the representations generated by: (a) DIFFPOOL; (b) SubGattPool, but the SubGatt embedding and pooling layers being replaced by GIN; (c) SubGattPool without intra and inter layer attention; (d) the complete SubGattPool network. Compared to (a), there are improvement of performances for both the SubGatt layer and intra/inter-level attention individually. Finally different classes are separated most by SubGattPool which again shows the merit of the proposed algorithm.



(a) (b) (c) (d)

Figure 5. Sensitivity analysis of SubGattPool for graph classification on MUTAG with respect to different hyper-parameters: (a) Maximum subgraph size, (b) Number of subgraphs sampled per epoch for each node, (c) Embedding dimension and (d) Number of SubGatt layers in SubGattPool.