Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Node classification using kernel propagation in graph neural networks

Sakthi Kumar Arul Prakash [a], Conrad S. Tucker [a,b,c,d,e,*]

[a] Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA
[b] Department of Machine Learning, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA
[c] The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA
[d] Department of Biomedical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA
[e] CyLab Security and Privacy Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA

ARTICLE INFO

Keywords:
Deep learning
Node classification
Network embedding
Graph neural networks
Attention

ABSTRACT

In this work, we introduce a kernel propagation method that enables graph neural networks (GNNs) to leverage higher-order network structural information without increasing the complexity of the networks. Recent studies have introduced GNNs that include higher-order neighborhood features containing global network information by propagating node features using a higher-order feature propagation rule. Though these GNNs have shown to improve node classification performance, they fail to include local connectivity information. Alternatively, GNNs also concatenate increasing orders of adjacency matrix in deeper layers in order to include higher-order structural information. In addition to global network information, GNNs also make use of node features which are network and node dependent features that serve to distinguish structurally isomorphic sub-structures within graphs. However, such node features may not always be available or depending on the network, may lead to deteriorating classification performance. Hence, to resolve these limitations, we propose a kernel propagation method that introduces a pre-processing step for GNNs to leverage higher-order structural features. The higher-order structural features are computed using a weighted random walk matrix which is node independent while using the first-order spectral propagation rule which explicitly considers local connectivity. Through our benchmark experiments, we find that the computed higher-order structural features are capable of replacing node dependent features while performing node classification task with performance on par with the state of the art approaches. Further, we also find that including both node features and higher-order structural features increases the performance of GNNs on large scale benchmark networks considered in this study. Our results show that considering local and global structural information as input to GNNs lead to an improvement in node classification performance in the absence/presence of node features without loss of performance.

1. Introduction

Graphs are ubiquitous data structures that allow relational knowledge about interacting entities to be efficiently stored, represented, and inferred. One of the principal problems in machine learning on graphs is finding an efficient way to represent information about the structure and homophily of the interacting entities into the machine learning models. Graph learning has enabled numerous graph analytic tasks such as community detection (Girvan and Newman, 2002; Rosvall and Bergstrom, 2008), link prediction (Lu and Getoor, 2003), node classification and clustering (Perozzi et al., 2014; Grover and Leskovec, 2016; Kipf and Welling, 2016; Li and Pi, 2019). Traditionally, most of the graph analytic tasks have relied on manually engineered feature vectors formulated with different graph statistics (Bhagat et al., 2011) such as clustering coefficient and degree measures or kernels (Gärtner et al., 2010) depending on the task. However, such approaches can be limiting in understanding the complex relationships between nodes, as graph data is usually high dimensional and hence may not be generalizable to all graphs. Recent advances in representation learning (Wu et al., 2020) using deep learning models have helped advance the performance of aforementioned tasks by constructing an efficient mapping for each node in a graph from a higher dimensional non-euclidean space to a meaningful low-dimensional vector representation.

Neighborhood aggregation methods use different aggregation techniques (Kipf and Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017) to aggregate feature representations from nodes in the graph to

* Corresponding author at: Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA.
E-mail addresses: sarulpra@andrew.cmu.edu (S.K. Arul Prakash), conradt@andrew.cmu.edu (C.S. Tucker).

https://doi.org/10.1016/j.eswa.2021.114655
Received 27 March 2020; Received in revised form 23 January 2021; Accepted 23 January 2021
Available online 4 February 2021

create efficient lower dimensional embeddings. However, such approaches pose risks of generalizability due to a dependence on the availability of meta-data such as node features. In the absence of node features, existing GNNs fail to differentiate between topologically similar graph sub-structures within graphs (Xu et al., 2018) and as a workaround, augment node features as one-hot encodings (Kipf and Welling, 2016; Veličković et al., 2017) or use deeper networks.

Previous work has focused on node features (Kipf and Welling, 2016; Veličković et al., 2017) which only provides a coarse representation of the network to capture position/location while using spectral and spatial approaches to compute structural features for node classification tasks. In addition, models trained using network-dependent node features may not generalize to other networks. Furthermore, neighborhood aggregation in baseline methods such as GCN and Graph Attention Networks (GAT) are unable to consider higher-order neighborhood information without increasing the number of neural network layers. However, increasing the depth of the models results in over-smoothing of node features (Li and Pi, 2019), wherein the representations of the nodes in the network converge to a stationary representation, further leading to the problem of vanishing gradients. Other approaches have looked at increasing the depth of GNNs with a top layer densely connected with earlier layers to concatenate/max-pool features (Xu et al., 2018), adapting neighborhood ranges (Klicpera et al., 2018) and using higher-order neighborhood (powers of adjacency matrix) aggregation techniques (Abu-El-Haija et al., 2018; Veličković et al., 2018) as a workaround to incorporate higher-order neighborhood representation learning objectives.

The aforementioned approaches improve classification accuracy by increasing the complexity of the model while assuming that there is a strong correlation between node features and network structure. Alternatively, since node features and structural features indeed describe the same network, we cannot neglect the correlation between them. Since node features characterize nodes, they may be used to differentiate between equivalent substructures. Hence, instead of following any one of the assumptions, we hypothesize that structural features and node-attribute features complement one another, while structural features are derived independently from the network without the help of node labels nor node features. Additionally, integrating the information on the connections of each node with the information about its features is crucial to discriminating essential and negligible characteristics of nodes (Bianconi et al., 2009), thereby minimizing false negatives and false positives.

To address the limitations posed by existing learning algorithms, we propose a kernel propagating graph neural network (KP-GNN) which makes the following contributions:

1. A pre-processing step that captures higher-order neighborhood information from a symmetric normalized adjacency matrix to compute a discounted t-hop diffusion affinity matrix that captures diffusion strength between nodes in a t-hop neighborhood. Such a feature representation captures higher-order structural information as an approximate limiting distribution of the normalized random walk matrix.
2. A kernel propagation technique that uses a first-order feature propagation rule which preserves target nodes' immediate neighborhood while propagating the pre-computed higher-order structural features and node features using a message passing neural network architecture.
3. A scaled Hadamard-product attention mechanism which reduces the number of hidden neurons/parameters by half in an attention head such as GAT while stabilizing the losses during training and validation. The use of the attention mechanism transforms KP-GNN to an attentional kernel propagating graph neural network (AKP-GNN).
4. An approach that shows node features can be replaced with higher-order structural features without loss of classification accuracy for all networks considered in this study. This contribution allows existing GNNs to be independent of node features when they are unavailable or missing while boosting the performance of GNNs when such features are available.

We validate the performance of AKP-GCN by comparing it with state of the art representative graph embedding methods, which includes node2vec (Grover and Leskovec, 2016), GCN (Kipf and Welling, 2016), GAT (Graph Attention Network) (Veličković et al., 2017), JK Net (Jumping Knowledge Network) (Xu et al., 2018), APPNP (Klicpera et al., 2018) and GDC (Graph Diffusion Convolution) (Klicpera et al., 1335). We test these methods using three benchmark small-scale real-world datasets and three benchmark large-scale real-world datasets on a node classification task. The experimental evaluation validates the performance of the proposed propagation technique.

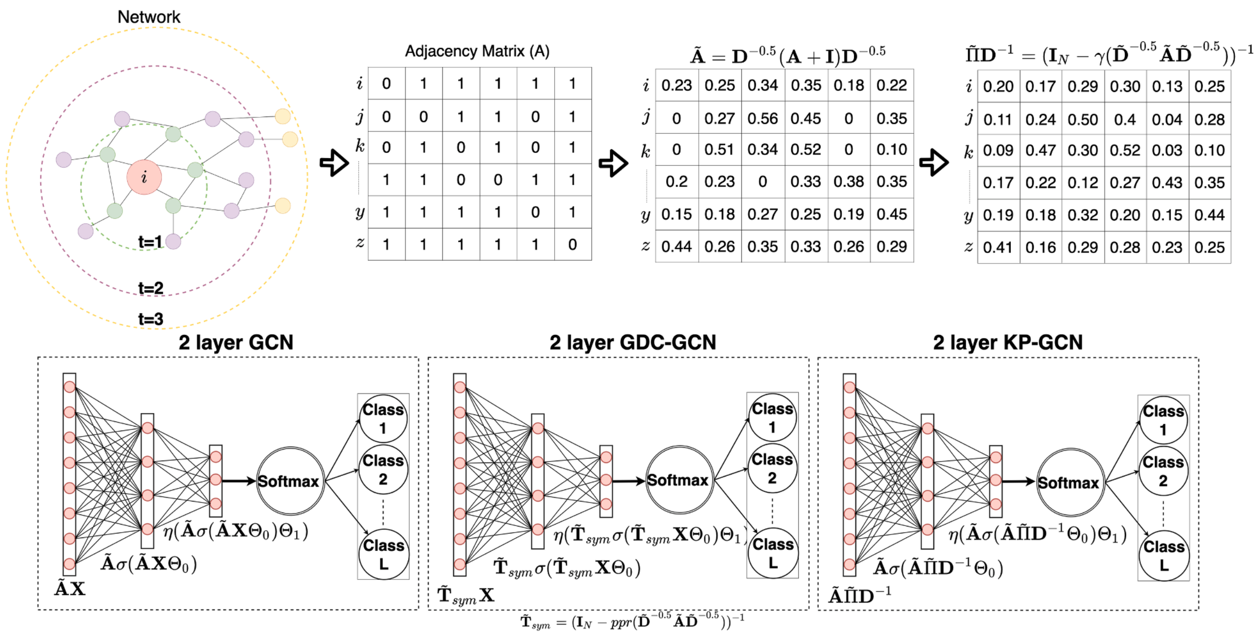
The remainder of the paper is organized as follows. This section provides an introduction and motivation for this work. Section 2 provides the background of related works. Section 3 provides preliminary concepts, definitions, and describes the proposed method in detail. Section 4 describes the statistics of the datasets, hyper-parameters for reproducing the work, methods being compared, followed by results and discussion. Section 5 concludes the paper.

2. Related work

2.1. Random-walk based approaches

Representation learning-based approaches embed the network onto a lower-dimensional vector space where information about the topology and communities of the network are preserved. The existing node representation algorithms can be classified as preserving the structural equivalence (topology) of a network, the homophily (similarity in communities) of a network, or a hybrid of both approaches (Grover and Leskovec, 2016). Homophily based methods include graph factorization (Belkin and Niyogi, 2002; Tang and Liu, 2011), structural equivalence methods include random walk and graph factorization (Henderson et al., 1450; Ribeiro et al., 1450) approaches, while hybrid methods (Perozzi et al., 2014; Grover and Leskovec, 2016; Qiu et al., 1450) incorporate objective functions capable of learning the network's homophily and topology with comparable computational efficiency. Other approaches in the literature include metric/modularity-based algorithms (Girvan and Newman, 2002; Newman and Girvan, 2004; Yang and Leskovec, 2015) and attributed graphs algorithms (Wang et al., 2016; Kipf and Welling, 2016; He et al., 2017; Veličković et al., 2017; Li et al., 2018) that are an augmentation of topology based methods wherein networks such as citation networks have nodes that contain attribute information also called *node features* that are important for downstream tasks such as node classification and link prediction.

Deepwalk (Perozzi et al., 2014), is an unsupervised deep learning method that belongs to the family of random-walk based embedding approaches. It learns latent network representations as a function of first-order proximity between nodes using the skip-gram model (Mikolov et al., 2013), which is a popular method first introduced in the domain of natural language processing to compute dense word embeddings which preserve similarity with similar words in a lower-dimensional embedding. However, Deepwalk was able to learn the inherent properties, such as first-order node proximity only. Node2vec (Grover and Leskovec, 2016) introduced a hybrid learning approach that combines the skip-gram (Mikolov et al., 2013) model with a neighborhood preserving likelihood objective using second-order Markovian random walks. Though an effective method, it uses several predefined hyper-parameters to choose between optimizing for network homophily or network structural equivalence. Since real-world networks exhibit varying levels of homophily and structural equivalence depending on the purpose of the network, it becomes an essential requirement to tune node2vec depending on the network and problem under study. Additionally, node2vec is a natural extension of Deepwalk such that the


Network
i
t=1
t=2
t=3
Adjacency Matrix (A)
$\tilde{\mathbf{A}} = \mathbf{D}^{-0.5}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-0.5}$
$\tilde{\Pi}\mathbf{D}^{-1} = (\mathbf{I}_N - \gamma(\tilde{\mathbf{D}}^{-0.5}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-0.5}))^{-1}$
2 layer GCN
$\tilde{\mathbf{A}}\mathbf{X}$
$\tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\mathbf{X}\Theta_0)$
$\eta(\tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\mathbf{X}\Theta_0)\Theta_1)$
Softmax
Class 1
Class 2
Class L
2 layer GDC-GCN
$\tilde{\mathbf{T}}_{sym}\mathbf{X}$
$\tilde{\mathbf{T}}_{sym}\sigma(\tilde{\mathbf{T}}_{sym}\mathbf{X}\Theta_0)$
$\eta(\tilde{\mathbf{T}}_{sym}\sigma(\tilde{\mathbf{T}}_{sym}\mathbf{X}\Theta_0)\Theta_1)$
Softmax
Class 1
Class 2
Class L
$\tilde{\mathbf{T}}_{sym} = (\mathbf{I}_N - ppr(\tilde{\mathbf{D}}^{-0.5}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-0.5}))^{-1}$
2 layer KP-GCN
$\tilde{\mathbf{A}}\tilde{\Pi}\mathbf{D}^{-1}$
$\tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\tilde{\Pi}\mathbf{D}^{-1}\Theta_0)$
$\eta(\tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\tilde{\Pi}\mathbf{D}^{-1}\Theta_0)\Theta_1)$
Softmax
Class 1
Class 2
Class L

Adjacency Matrix (A)

i	0	1	1	1	1	1
j	0	0	1	1	0	1
k	0	1	0	1	0	1
⋮	1	1	0	0	1	1
y	1	1	1	1	0	1
z	1	1	1	1	1	0

$\tilde{\mathbf{A}} = \mathbf{D}^{-0.5}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-0.5}$

i	0.23	0.25	0.34	0.35	0.18	0.22
j	0	0.27	0.56	0.45	0	0.35
k	0	0.51	0.34	0.52	0	0.10
⋮	0.2	0.23	0	0.33	0.38	0.35
y	0.15	0.18	0.27	0.25	0.19	0.45
z	0.44	0.26	0.35	0.33	0.26	0.29

$\tilde{\Pi}\mathbf{D}^{-1} = (\mathbf{I}_N - \gamma(\tilde{\mathbf{D}}^{-0.5}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-0.5}))^{-1}$

i	0.20	0.17	0.29	0.30	0.13	0.25
j	0.11	0.24	0.50	0.4	0.04	0.28
k	0.09	0.47	0.30	0.52	0.03	0.10
⋮	0.17	0.22	0.12	0.27	0.43	0.35
y	0.19	0.18	0.32	0.20	0.15	0.44
z	0.41	0.16	0.29	0.28	0.23	0.25

Fig. 1. Proposed Architecture for combining higher-order structural features with first-order feature propagation scheme. Each dotted circle within the network represents the *t*-hop neighborhood of node *i*. The figure provides a visual flow of the data from the network form.

walks learn the local and the global neighborhood transition of a node as features.

2.2. Attention and neighborhood aggregation methods

Attention mechanisms were first introduced in sequence to sequence learning in the domain of machine translation by Bahdanau et al. (2014) and Luong et al. (2015.) to focus on different parts of the input sequence at every stage of the output sequence, thereby allowing context information to be preserved from beginning to end. Xu et al. (2015) used attention for image captioning tasks, while Mnih et al. (2014) applied attention for image classification tasks. Although attention mechanisms have been successfully applied to many problems, most of the existing work pertains to the domains of computer vision or natural language processing. Though graphs have an irregular structure which is non-Euclidean when compared to images/videos (grids) but sequential such as text data, attention became readily applicable to graph data with modifications to the traditional definition of attention mechanisms to fit GNNs (Lee et al., 2019).

Neighborhood aggregation algorithms leverage node features and attention mechanisms to generate better node representations. Current GNN models are variants of message-passing architectures that use various neighborhood aggregation schemes to aggregate feature information from nodes in the graph wherein such architectures can either be spatial (Veličković et al., 2017; Hamilton et al., 2017) or spectral (Kipf and Welling, 2016). Graph Convolutional Networks (Kipf and Welling, 2016) uses a spectral rule that employs graph convolutional layers to learn node embeddings by aggregating one-hop local graph structures and features of nodes to obtain embeddings from the hidden layers; while Graph Attention Networks (Veličković et al., 2017) being a spatial approach, aggregates neighborhood information according to trainable attention weights using multi-head attention mechanisms. These models focus on learning node representations that are capable of capturing one-hop local network structure using a single hidden layer. Hence, these message passing neural networks, both spectral and spatial are not capable of leveraging higher-order structural features without deeper layers. However, increasing the number of layers in GCN or GAT has shown to deteriorate their performance (Li et al., 2018).

While spectral methods have nice theoretical properties, MPNNs such as HO-GCN (Abu-El-Haija et al., 2018) and GraphSAGE (Hamilton et al., 2017) increase the neighborhood range from one-hop and make use of higher-order convolution layers or higher-order neighborhood aggregation techniques (Xu et al., 2018; Klicpera et al., 2018) to outperform spectral based methods. This however increases model complexity in order to achieve higher node classification performance while assuming that there is a strong correlation between node features and network structure. Concurrently, neglecting the existence of any correlation between the structural and node attribute features is also a strong assumption that may not hold true in all cases. Further, work such as Approximate Personalized Propagation of Neural Predictions (APPNP) (Klicpera et al., 2018) and Graph Diffusion Convolution(GDC) (Klicpera et al., 1335) considers higher-order neighborhood information by computing a stationary distribution pertaining to the random-walk matrix as well as the importance of the neighborhoods by incorporating a weighting scheme based on Personalized PageRank (PPR) and heat kernel (Chung, 2007; Donnat et al., 2018; Klicpera et al., 2018). These approaches however lose focus of the immediate first-order neighborhood of a target node by using a higher-order feature propagation scheme that only considers global connectivity. Though APPNP uses a teleport vector with no spectral properties to preserve local connectivity, their final expression for the propagation scheme is similar to that of a limiting distribution of a random walk matrix. Additionally, the weighting schemes used in these approaches were originally developed to approximate a graph partitioning algorithm or specifically local graph partitioning (Chung, 2007) which requires constraints and predefined partition sizes.

In this work, we first assume that high-order structural and node features are complementing. Second, we derive a kernel propagation technique that computes an optimum limiting diffusion kernel as a normalized discounted geometric series of the random walk matrix of the network. The pre-processed higher-order structural features are then propagated using an explicit first-order spectral definition. Further, the higher-order structural features are also augmented with node features which capture network-dependent node characteristics relevant for discriminating structurally equivalent structures. Such a technique is capable of capturing network and node independent features since the proposed discounted geometric approximation is a variation of the weighting scheme used in capturing community structure from real-

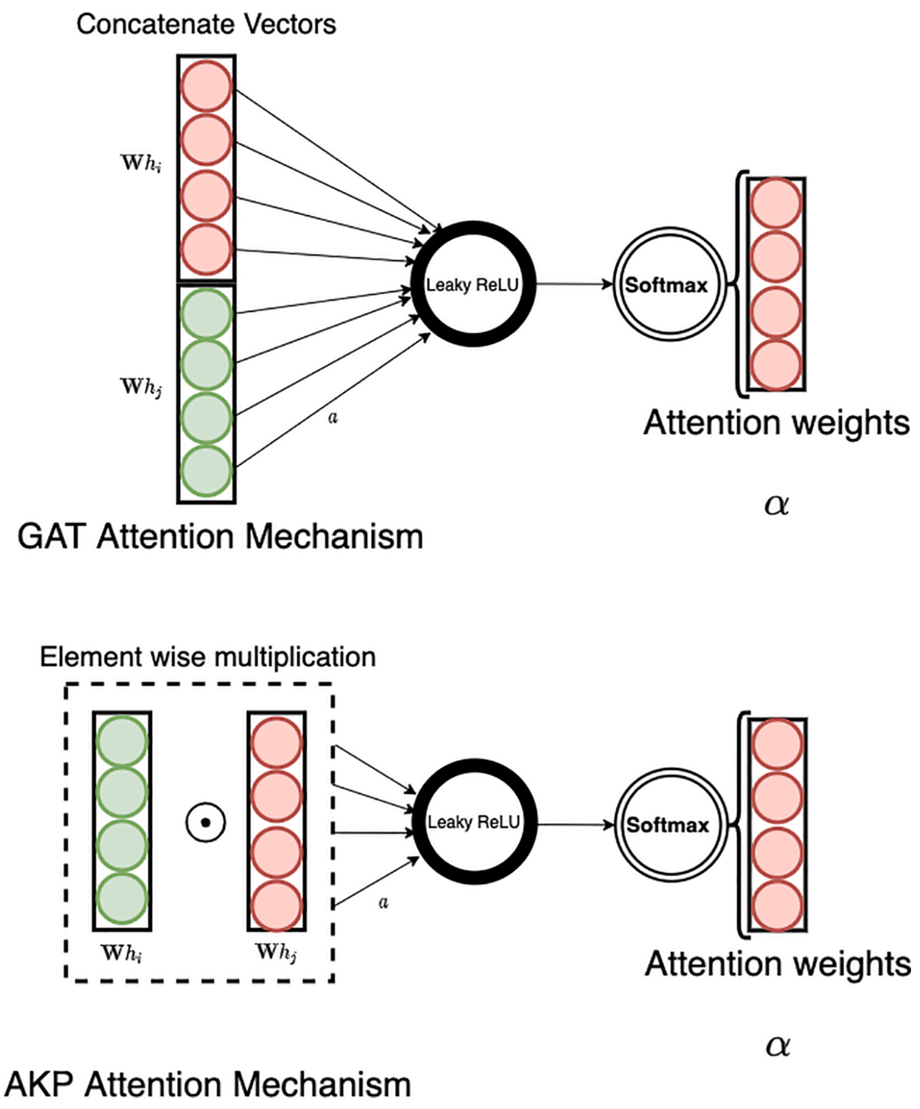


Fig. 2. Difference between the attention mechanism used in GAT and the proposed AKP.

world networks (Girvan and Newman, 2002), thus enabling node classification performance on par with the state of the art methods. Hence, this motivates the need for hybrid GNN models that are capable of combining network independent and network-dependent features, as it is clear that position aware (local and global) approaches are necessary (Xu et al., 2018; You et al., 2019) to improve the performance of GNNs in link prediction and classification tasks.

3. Kernel propagation technique

3.1. Notations

A network $\mathscr{G} := \{\mathscr{V}, \mathscr{E}\}$ is an order 2 Tensor such that $\mathscr{G} \in \mathbb{R}^{N\times N}$ where N represents the number of users or $\mathscr{V}$ such that $\mathscr{V} \in \mathbb{R}^{N}$ and $\mathscr{E} := \{(i,j)|i,j \in \mathscr{V} \times \mathscr{V}, i \neq j, \mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}\}$ is the set of all pairs of distinct nodes, called edges. For every node $i \in \mathscr{V}$, the degree $d(i)$ is the number of edges leaving or entering i, such that $d(i) = \sum_j A_{ij}$, where $A_{ij} \in \mathbf{A}$ and $\mathbf{A}$ denotes the adjacency matrix while $\mathbf{D}$ denotes the degree matrix of $\mathscr{G}$. We introduce $\mathscr{L}$ which denotes the semi-supervised loss with respect to the labeled part of the graph and $f(.)$ be a neural network. Let $\mathbf{X}$ denote the node features matrix. We define $\mathbf{H}^{(l)} = [h_1^{(l)}, h_2^{(l)}, \ldots, h_N^{(l)}]$ as the hidden layer representation of the network in the l^{th} layer of $f(.)$ where $h_i^{(l)} \in \mathbb{R}^N$ represents node i. Fig. 1 presents an overview of our proposed method for learning node representations from a network $\mathscr{G}$ and performing node classification in a complete end to end manner. The figure depicts an example of how node i's representation is computed and is later used for node classification. The following sections describe the method in detail.

3.2. Kernel propagation layer

Following (Kipf and Welling, 2016), we define the following layer-wise propagation technique:

$$\mathbf{H}^{(l+1)} = \sigma\left(\widetilde{\mathbf{D}}^{-0.5}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-0.5}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{1}$$

In Eq. (1), $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N, \widetilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}_N$, where, $\mathbf{I}_N$ is an identity matrix of shape (N,N), $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{A}_{ij}$, $\sigma(.)$ denotes a non-linear activation function such as ReLU(.), $\mathbf{H}^{(0)} = \mathbf{X}$ and $\mathbf{W}^{(l)}$ is a layer-specific trainable weight matrix. The graph convolutional layer above represents each node i as an aggregate of its neighborhood $\mathscr{N}_i$ such that it includes itself as a result of self-loops.

Given the adjacency matrix of the network, we compute the random walk transition matrix which provides information on the probability of node transitions within the network. The random walk transition matrix $(\mathbf{P})$ represents the probability of a node (i) diffusing information or forming an edge with another node (j) given the previous state of the nodes where $\mathbf{P} = P_{ij} = Pr(j|i)$ is a conditional probability which represents the probability of a node (i) transitioning from i to j in one step. In our proposed update rule, we enable feature augmentation by concatenating the node features matrix ($\mathbf{X}$) with an optimal t-hop diffusion affinity matrix/kernel Π^* which is a network independent feature matrix derived from network topology (adjacency matrix). We define random walk transition matrix as $\mathbf{P}_{rw} = \mathbf{AD}^{-1}$. Since the graph convolutional layer uses a normalized symmetric adjacency matrix with self loops, we calculate transition matrix using $\widetilde{\mathbf{A}}$ as $\widehat{\mathbf{P}} = \widetilde{\mathbf{D}}^{-0.5}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-0.5}\widetilde{\mathbf{D}}^{-1}$. We derive the expression for Π, the diffusion kernel as follows:

$$\Pi = \sum_{t=0}^{\infty} \widehat{\mathbf{P}}^t \tag{2}$$

Expanding Π, we rewrite Eq. (2) as $(\mathbf{I}_N + \widehat{\mathbf{P}}^1 + \widehat{\mathbf{P}}^2 + \ldots + \widehat{\mathbf{P}}^\infty)$, which can further be simplified as $(\mathbf{I}_N - \widehat{\mathbf{P}})^{-1}$ using the result of the geometric series where, $\widehat{P}_{ij} < 1$. We further normalize (symmetric normalization can also be performed) the expression $(\mathbf{I}_N - \widehat{\mathbf{P}})^{-1}$ as $(\mathbf{I}_N - \widehat{\mathbf{P}})^{-1}\widehat{\mathbf{D}}^{-1}$ where, $\widehat{\mathbf{D}}^{-1}$ is the degree matrix of the expression $(\mathbf{I}_N - \widehat{\mathbf{P}})^{-1}$. However, computing $(\mathbf{I}_N - \widehat{\mathbf{P}})^{-1}$ poses a serious drawback in that, it may not be possible to compute the inverse of matrix with huge number of nodes without significant computing resources. Hence, as a workaround, we compute an optimal Π^* in just a few exponentials by introducing a discount factor γ such that $\gamma \in [0,1]$ and the expression we are trying to compute above is modified as $(\mathbf{I}_N - \gamma\widehat{\mathbf{P}})^{-1}$. This expression, unlike personalized page rank (PPR), does not take into consideration the root node and assumes no hierarchical node ordering in terms of node importance. Instead, γ weights all paths (node independent and node dependent paths) of a particular path length t by a factor γ^t.

From Lemma 1 (see appendix for proof), and since $\gamma < 1$, we know $\|\mathscr{M}^{(t)}(\widetilde{\Pi}) - \mathscr{M}^{(t)}(\Pi)\|_\infty \leqslant \gamma^{(t)}\|\widetilde{\Pi} - \Pi\|_\infty$, thus converging to a limit Π^* for $t \ll \infty$. Given that we have derived a converging limit Π^*, we now state the kernel (because of the similarity of $(\mathbf{I}_N - \gamma\widehat{\mathbf{P}})^{-1}$ with regularized graph Laplacian kernel (Smola and Kondor, 2003)) propagation technique as follows.

$$\mathbf{H}^{(l+1)} = \sigma\left(\left(\widetilde{\mathbf{D}}^{-0.5}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-0.5}\right)\left(\left[\Pi^*\widehat{\mathbf{D}}^{-1}, \mathbf{X}\right]\right)\mathbf{W}^{(l)}\right) \tag{3}$$

Following equation Eq. (1), we augment $\mathbf{H}^{(0)}$, which is the node features matrix ($\mathbf{X}$) with the normalized t-hop diffusion kernel, when $l = 0$. The concatenated feature matrix re-weights the normalized symmetric adjacency matrix, using node features and higher-order neighborhood information. In the next section, we describe the multiplicative aggregation method which aggregates the hidden representations of the nodes such that nodes that belong to the same community are closer in the embedding space.

3.3. Multi-head multiplicative attention mechanism

The output from the kernel propagation layer produces a new set of node representations $\mathbf{H}^{(l+1)} = [h_1^{(l+1)}, h_2^{(l+1)}, \ldots, h_N^{(l+1)}]$ where, $h_i^{(l+1)} \in \mathbb{R}^{\widetilde{N}}$ represents node i's features in the $(l+1)$ layer. Attention mechanism in its general formulation allows every node representation to attend on every other node's representation. To preserve graph structure in the node representations computed from the propagation step, we perform a masked shared attentional mechanism (Veličković et al., 2017) $a : \mathbb{R}^{\widetilde{N}} \text{X}\ \mathbb{R}^{\widetilde{N}} \longrightarrow \mathbb{R}$ following the modified GCN layer, which is parametrized by a weight matrix, $\mathbf{W}^{(l+1)} \in \mathbb{R}^{\widetilde{N}xN}$. In our work, instead of applying a to the concatenated representations of i and j, we employ a multiplicative Hadamard product attention. This decreases the number of hidden layer parameters to be learned by 50%. Fig. 2 presents an illustration of the difference between the attention mechanism employed in this paper and GAT. Following the transformations, the importance of node j's hidden representation to node i's representations is captured as follows,

$$e_{ij} = a\left(\mathbf{W}h_i^{(l+1)} \odot \mathbf{W}h_j^{(l+1)}\right) \tag{4}$$

We compute e_{ij} for nodes j which are in the neighborhood ($\mathscr{N}_i$) of node i. In our experiments, the neighborhood ($\mathscr{N}_i$) will be the first-order neighbors of i, including i. The shared attention mechanism a is a single-layer feedforward neural network, parametrized by a such that $a \in \mathbb{R}^{\widetilde{N}}$ which is then subject to a non-linear activation function $\in(.)$ such as LeakyReLU(.). However, the Hadamard product of the representations need to be scaled in order to avoid exploding products, which in turn makes softmax gradients extremely small (Vaswani et al., 2017). To counteract this effect, the Hadamard products are scaled using $\sqrt{K}$ as shown in Eq. (5) where, K represents the number of heads used in computing the attention coefficients. Following the LeakyReLU non-linear activation ($\in(.)$), the attention coefficients computed by the attention mechanism are expressed as:

$$\alpha_{ij} = \frac{\exp\in\left(\dfrac{\mathbf{W}h_i^{(l+1)}\left(\mathbf{W}h_j^{(l+1)}\right)^T}{\sqrt{K}}\right)}{\sum_{c\in\mathscr{N}_i}\exp\in\left(\dfrac{\mathbf{W}h_i^{(l+1)}\left(\mathbf{W}h_c^{(l+1)}\right)^T}{\sqrt{K}}\right)} \tag{5}$$

The normalized attention coefficients as computed using Eq. (5) are then linearly combined with respect to each node's hidden representation from layer (l) after applying a non-linearity $\eta(.)$ such as Exponential Linear Unit (ELU) as shown below.

$$\widetilde{h}_i^{(l+1)} = \eta\left(\sum_{j\in\mathscr{N}_i}\alpha_{ij}\mathbf{W}h_j^{(l)}\right) \tag{6}$$

Additionally, we employ multi-head attention similar to Vaswani et al. (2017) to stabilize the learning process of self-attention. In our experiments, we implement K independent attention mechanisms.

3.4. Semi-supervised node classification

Having introduced the kernel propagation layer and the multi-head attentional layer, we define the KP/AKP-GNN model as $f(\widetilde{\mathbf{A}}, \Pi^*)$ and address the problem of semi-supervised node classification by training our model using task-specific loss function. The complete forward propagation of our two layer model with a softmax activation on the final layer is defined as follows:

$$\widehat{\mathbf{Y}} = f\left(\widetilde{\mathbf{A}}, \Pi^*\right) = softmax\left(\alpha\left(\sigma\left(\left(\widetilde{\mathbf{D}}^{-0.5}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-0.5}\right)\left(\left[\Pi^*\widehat{\mathbf{D}}^{-1}, \mathbf{X}\right]\right)\mathbf{W}^{(0)}\right)\right)\mathbf{W}^1\right) \tag{7}$$

where, $\mathbf{W}^0 \in \mathbb{R}^{(N+C)x\widetilde{N}}, \mathbf{W}^1 \in \mathbb{R}^{(Kx\widetilde{N})x\omega}$, C is the number of column dimensions of a node's feature vector ($\mathbf{X}_i$) and ω is the number of final hidden layer neurons. The loss function ($\mathscr{L}$) for the multi-class node classification task is evaluated as the cross-entropy error over the labelled examples as follows:

$$\mathscr{L} = -\sum_{i\in\mathscr{V}^L}\sum_{s=1}^{S}\left[Y_{is}\ln\widehat{Y}_{is}\right] \tag{8}$$

Here, $\mathscr{V}^L$ is the set of nodes in the training data for which class labels are available, while S denotes the number of classes such that $s \in S$, $\mathbf{Y}$ is the matrix of ground-truth one hot encoding denoting the class assignment per node and $\widehat{\mathbf{Y}}$ is the final layer predictions of the AKP-GCN model. The network weights are trained end-to-end using the gradient descent

Table 1
Network Statistics and training/validation split (Multi-class, Single-label (MC, SL), Multi-class, Multi-label (MC,ML).

	Cora	Citeseer	PubMed	ogbn-proteins	ogbn-arxiv	ogbn-products
Nodes	2708	3312	19,717	132,534	169,343	2,449,029
Edges	5429	4732	44,324	39,561,252	1,166,243	61,859,140
Node/edge features	1433	3703	500	8	128	100
Classes	7	6	3	112	40	47
Classification type	MC, SL	MC, SL	MC, SL	MC, ML	MC, SL	MC, SL

algorithm.

4. Experiments and discussion

We validate the feasibility of our algorithm against widely used datasets by performing a multi-class, single-label classification to validate the performance of the proposed algorithm in classifying nodes based on the learned node representations. This section introduces the experimental settings, followed by experimental results, and finally discusses the results. The statistics of the datasets used in this work are given in Table 1.

4.1. Experimental settings

4.1.1. Baseline methods

The choice of methods to benchmark against was motivated by publication year, relevance to the proposed method, and current state of the art. We benchmark our method with the following:

- Node2vec (Grover and Leskovec, 2016): This method also belongs to the family of random-walk based approaches and generalizes the Deepwalk approach. Node2vec performs a second-order proximity learning, where random walks are traversed breadth-wise and depth-wise, unlike Deepwalk which performs a 1st order proximity learning.
- GCN (Kipf and Welling, 2016): GCN is one of the most popular GNN models with most of the recent work and applications being built on top of this architecture. The spectral rule as proposed by GCN can represent a one-hop node neighborhood with a single layer and requires additional layers for aggregating higher-order node neighborhoods. Our method is a natural variant of the GCN.
- GAT (Veličković et al., 2017): GAT is an extension of GCN with an aggregation based methodology, which employs self-attention with weight sharing to place similar node representations from one-hop neighborhoods closer together in the embedding space.
- JK-Net (Xu et al., 2018): This work advances the use of deep graph neural networks, especially GCN which typically suffers from over-smoothing with deeper layers (Oono and Suzuki, 2019). The authors propose an architecture that selectively combines different aggregations at the final layer – jumping knowledge (JK) networks.
- APPNP (Klicpera et al., 2018): This work advances the use of higher-order neighborhood information in a forward propagation scheme where the neighborhood is weighted using a heat kernel or PPR coefficients. Further, this approach disentangles the propagation technique from the feature predictions scheme thereby enabling the use of deeper layers without being susceptible to over-smoothing.
- GDC (Klicpera et al., 1335): The graph diffusion convolution approach proposes to remove the restriction of only considering the first-order neighborhood through the introduction of a generalized graph diffusion. GDC is a GNN that combines the strengths of spatial and spectral methods. This approach deviates from that of the proposed approach since we consider higher-order structural information as features and further uses a general weighting scheme that captures structural information as node independent features.

4.1.2. Network statistics

We compare the methods described in sub-Section 4.1.1 by comparing the node classification accuracies on commonly used citation networks such as Cora, Citeseer and PubMed. In addition to the small-scale networks, we benchmark and evaluate the proposed approach and the comparing approaches on large-scale real-world networks published in the open graph benchmark (ogb) project (Hu et al., 2020). In this paper, since the problem under study is a node classification problem, we use the ogb-proteins, ogb-arxiv and the ogb-products datasets to benchmark the proposed approach. These datasets were chosen based on the nature of the networks (homogeneous graphs), and the size of the network (so as to fit the entire graph in the GPU memory) without resorting to batching techniques since they are not the focus of this study. Further, nodes in all the networks under study have node features, except for the ogb-proteins network which contains edge features. The dataset statistics are summarized in Table 1. For the small-scale real world networks under study, we use 15% of the nodes for training, 500 nodes for validation and testing. For the networks from the ogb project, we use the same training, validation and test split as used in their paper. In the results section, we report the average results of 3 runs for each of the methods.

4.1.3. Cora, citeseer and PubMed networks

In the experiments that use the small-scale networks, we set the number of KP-GCN and AKP-GCN hidden layers as one. The number of hidden units in the convolution layer is set as 64 for Cora and Citeseer, and 128 for PubMed. For the attentional layer, we use 4 attention heads for Cora, Citeseer and PubMed datasets. The hyper-parameters for our models are set as follows: dropout rate = 0.5, alpha value for LeakyReLU as 0.2, weight decay (L2 Norm) as $5e^{-4}$ and discount factor (γ) as 0.99. We use Xavier initialization (Glorot and Bengio, 2010) for all weight and attention matrices. For the optimization, we use Adam (Kingma and Ba, 2014) with a fixed learning rate of 0.02 and set the number of training epochs as 50.

4.1.4. OGB networks

In the experiments which use the ogb networks, we set the number of hidden layers as 2 and the number of attention heads as 4. The number of hidden units in the final layer is set as 128. Due to GPU memory constraints, we were unable to implement the attention mechanism on the ogb-proteins and ogb-products networks. The remaining hyper-parameters for all models are set as follows: dropout rate $=0.5$, attention dropout $=0.0$, alpha value for LeakyReLU $=0.1$, discount factor $(\gamma)=0.99$ and training epochs per run $=200$.

4.2. Experimental results and discussion

4.2.1. AKP-GCN models

In order to contrast and discuss the contributions and advantages of the proposed method, we introduce multiple versions of our model as follows:

1. **KP-GCN** using structural features and/or node features and no attention mechanism for a GCN architecture.
2. **AKP-GCN** using structural features and/or node features and proposed attention mechanism for a GCN architecture.

Table 2
Computational complexity of proposed models.

	Computational complexity
KP-GCN	$\mathscr{O}(N^3) + \mathscr{O}(\left\|\mathscr{E}\right\|(N+C)\widetilde{N}\omega)$
AKP-GCN	$\mathscr{O}(N^3) + \mathscr{O}(\left\|\mathscr{E}\right\|(N+C)\widetilde{N}\omega) + \mathscr{O}(N^2\omega + \left\|E\right\|\omega)$
KP-GAT	$\mathscr{O}(N^3) + \mathscr{O}(N(N+C)\widetilde{N}\omega + \left\|E\right\|\widetilde{N}\omega)$
KP-JK Net	$\mathscr{O}(N^3) + \mathscr{O}(\left\|\mathscr{E}\right\|(N+C)\widetilde{N}\omega)$

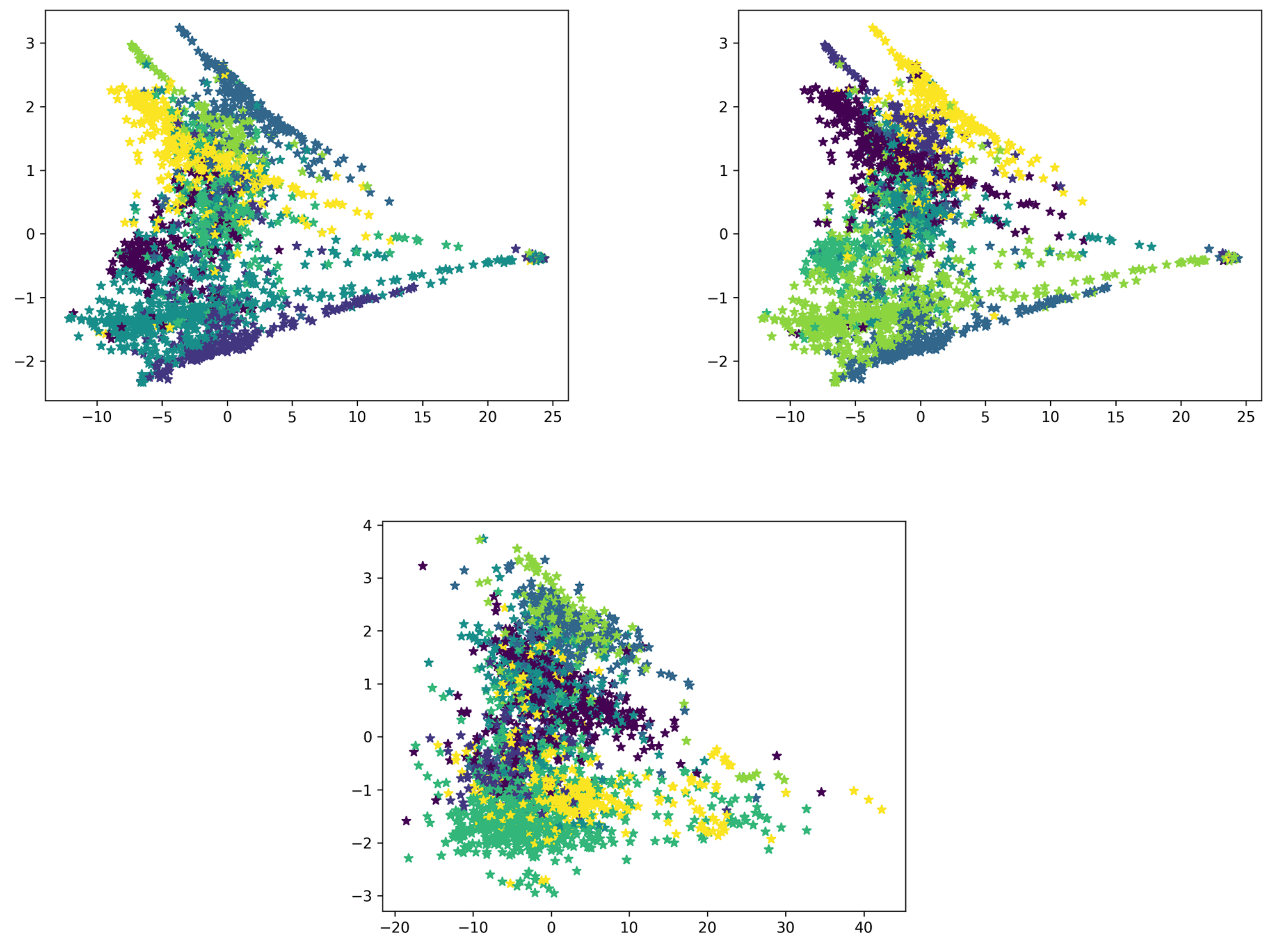


Fig. 3. (a) Learned Cora embedding using structural features and attention (b) Learned Cora embedding using node features and attention (c) Learned Cora embedding using structural features, node features and attention.

3. **KP-GAT** using structural features and/or node features and no attention mechanism for a GAT architecture.
4. **KP-JK Net** using structural features and/or node features and no attention mechanism for a JK Net architecture.

We have publicly made available the official version of the codebase pertaining to this work in GitHub and have ensured computational reproducibility using Code Ocean. Interested readers can refer to the following resources.[1,2]

Table 2 illustrates the computational time complexity for each of the models (2 layers) proposed above. Since the pre-processing step computes a t-hop power series of a transition matrix, the time complexity is $\mathscr{O}(N^3)$. Since we did not modify the internal architecture of GCN, GAT and JK Net, we use the computational time complexities reported in the respective papers while modifying the variables depending on the size of the input and that of the intermediate steps. The computational complexity reported for attention based model is for one attention head.

4.3. Parameter settings

4.3.1. Visualization and convergence analysis

Figs. 3–5 visualize the learned network embeddings for Cora, Citeseer and PubMed networks when using attention, structural and node attribute information. From the learned embeddings, it is evident that learning with node features or learning using the proposed diffusion kernel produces a similar discrimination of the different classes in a 2 dimensional feature space, while when concatenated together, produces a different discrimination in the case of Cora and Citeseer networks as shown by Figs. 3c and 4c. We omit visualizing the other three data sets since there exists far too many classes and nodes to visualize the discrimination between classes.

Figs. 6 and 7 illustrates the convergence of the proposed models trained using Cora network, where the proposed attention mechanism helps prevent model over-fitting (training loss is significantly lower than validation loss) during training, leading to a stable model convergence. Further, we find that the generalization gap between validation and testing accuracies for Cora, Citeseer and PubMed to be negligible when AKP-GCN model is utilized for training and testing. Additionally, we also observe that concatenating structural information along with node-attribute information helps the model optimize an easier loss

[1] https://github.com/AiPEX-Lab/Kernel-Propagation-in-Graph-Neural-Networks.

[2] https://codeocean.com/capsule/4740198/tree.

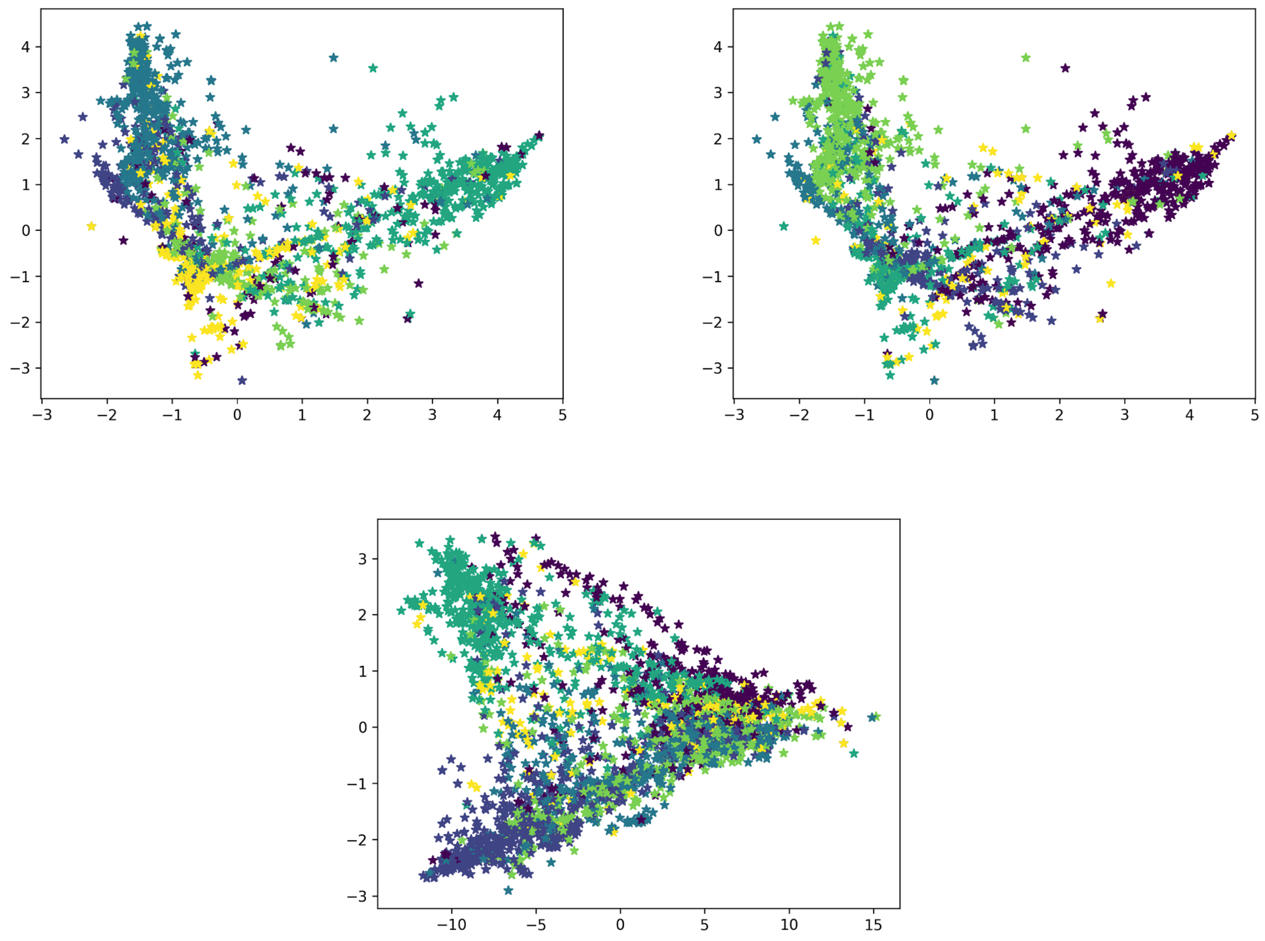


Fig. 4. (a) Learned Citeseer embedding using structural features and attention (b) Learned Citeseer embedding using node features and attention (c) Learned Citeseer embedding using structural features, node features and attention.

landscape (starting accuracy tends to be high, loss tends to be low) leading to a better model fit.

However, for the other three datasets, namely the ogbn-proteins, ogbn-arxiv and ogbn-products, we observe and report the existence of a generalization gap (as defined in Hu et al. (2020)) while using the proposed models as well as the comparing approaches. In the next section, we quantify the accuracies and discuss the merits of the various models used in the study.

4.3.2. *Comparison between methods and discussion of experimental results*

Table 3 displays the classification accuracy of the various methods on the citation networks. Our method achieves superior classification performance in comparison to the other methods. The kernel propagating layer in the absence of node features and with/without attention mechanism achieves comparable accuracy with respect to other methods and model variations, while the concatenated structural and node attribute features with/without attention mechanism achieves state of the art performance on the Cora and PubMed networks. Though we find that structural features and node attribute features complement each other, when node features are unavailable, network-independent structural features can be used in their place without loss in accuracy. However, the node-features only model performed node classification with comparable accuracy on the Citeseer network against our model which uses node-features and structural features. Hence, we observe that networks have an influence on how much information is encoded as topology versus node features. In the case of the Cora network, structural features can improve node classification performance when combined with node features. A similar case is observable for the PubMed network as well. A possible reason for this scenario is the lower nodes to edges ratio in the Cora and PubMed networks as well as the number of node features being fewer than the number of nodes in the Citeseer network. Hence node features enable better classification in the Citeseer network which seems to present more importance to the behavior of the nodes as opposed to node links while node features and structural features enable better discrimination of nodes in the other two networks. Further, by using the PubMed network where the number of nodes is of an order of magnitude more than the Cora and Citeseer networks, we show that our propagation technique is scalable to large networks.

The small-scale citation networks have been a topic of criticism within the representation learning community since their network sizes are not comparable to real-world networks that are orders of magnitude larger and realistic. Hence, to validate the performance of the proposed approach on existing GNNs, we perform experiments on large-scale ogb datasets. Further, as noted by Hu et al. (2020), we find the training,

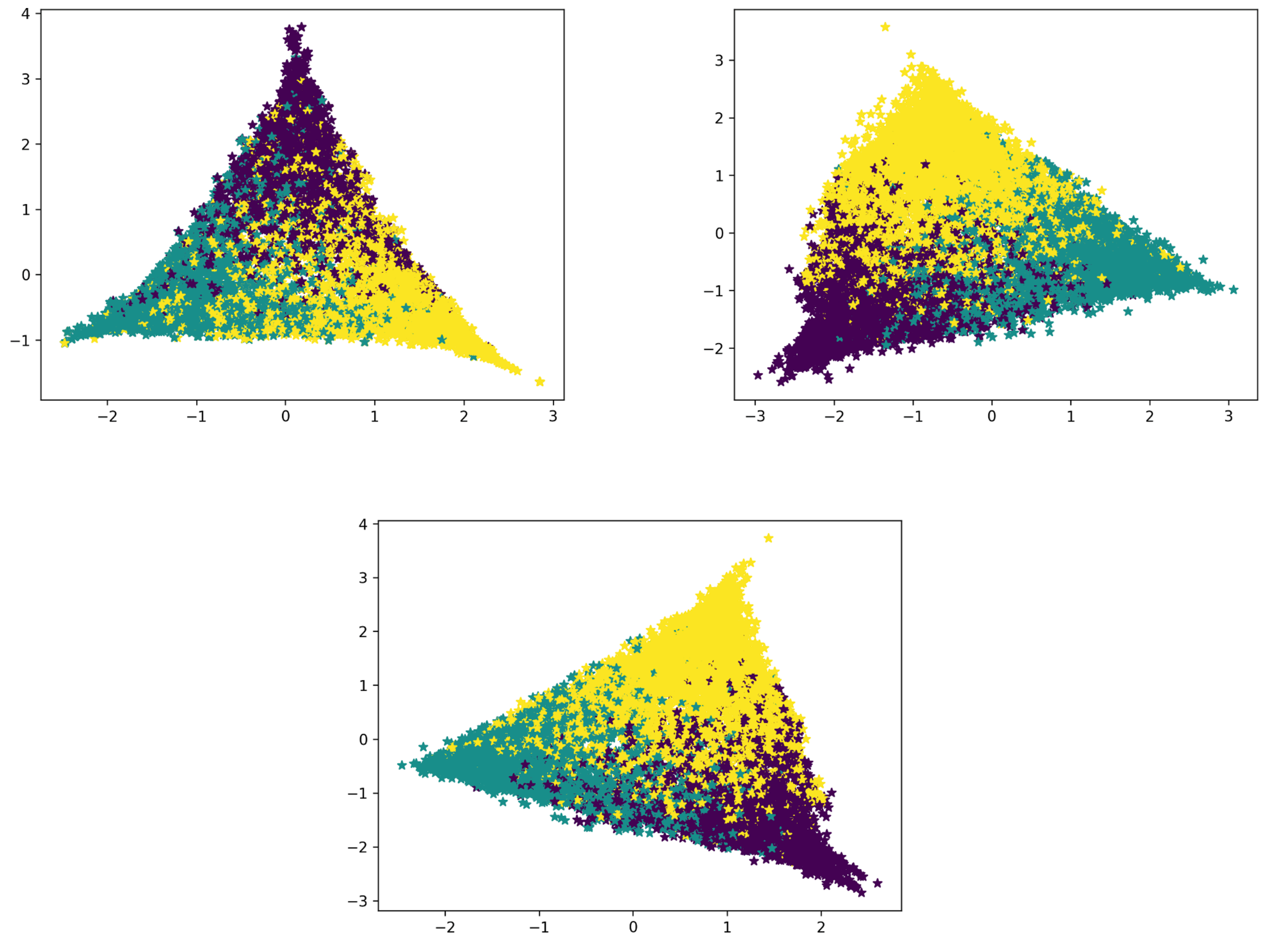


Fig. 5. (a) Learned PubMed embedding using structural features and attention (b) Learned PubMed embedding using node features and attention (c) Learned PubMed embedding using structural features, node features and attention.

validation and testing splits of the ogb datasets to be realistic and suitable for real-world deployment. In Table 4, we compare the validation and training accuracies of the pre-processing step on the ogb-arxiv network using GCN, GAT and JK Net variants using our proposed approach. In our experiments, we test our proposed approach against the baseline methods with node features (denoted by **X**) and without node features (denoted by **I** which is an identity matrix that replaces **X**). We find our KP-GAT model to outperform other approaches by a margin of approximately 0.4% on average, and further note that our models are able to improve the node classification accuracy of GCN, GAT and JK Net. When compared to GDC which is also a pre-processing step, our approach improves the node classification accuracy by a significant margin. This highlights the inherent problem of lost focus of one-hop neighborhood connectivity since the propagation scheme implemented in GDC is of higher-order. Further, we notice that our KP-GAT implementation which makes use of only higher-order neighborhood structure outperforms GCN, GAT and JK Net when no node features are considered. This highlights the ability of the proposed approach to identify meaningful structural information when node features are unavailable. We also note that including node features further improves the performance of our approach applied to GNNs.

Now we consider the ogb-products network, which is a sales network and is also the largest network considered in this study. Given the size of the network, we were unable to fit any attention models in the GPU and found training using CPU to be significantly time-consuming (approximately 3–4 days to run 200 epochs per run for a total of three runs per GNN variant). We were unable to find any GAT results from the ogb paper (Hu et al., 2020) for this dataset. We report our results in Table 5. We compare the validation and training accuracies of our approaches with the baseline methods. We report and compare the results of our approach applied to multiple GNN variants with the baseline approaches while considering node features (denoted by **X**) and excluding node features (denoted by **I** which is an identity matrix that replaces **X**). We find our KP-JK Net ($[\mathbf{X}, \Pi^{*}\widehat{\mathbf{D}}^{-1}]$) model outperforms other approaches by a margin of approximately 1.0% on average. We further note that our approach can improve the node classification accuracy of GCN and JK Net by approximately 3.5% on average when considering node features and higher-order structural features. Further, we note that GCN and APPNP perform significantly better when node features are not utilized during training. This shows that the node features pertaining to the ogb-products network could be complex, leading to no learning, as opposed to just learning self-attention. The training results further emphasize the rich implicit higher-order structural information of the network which can replace node features even when they are available.

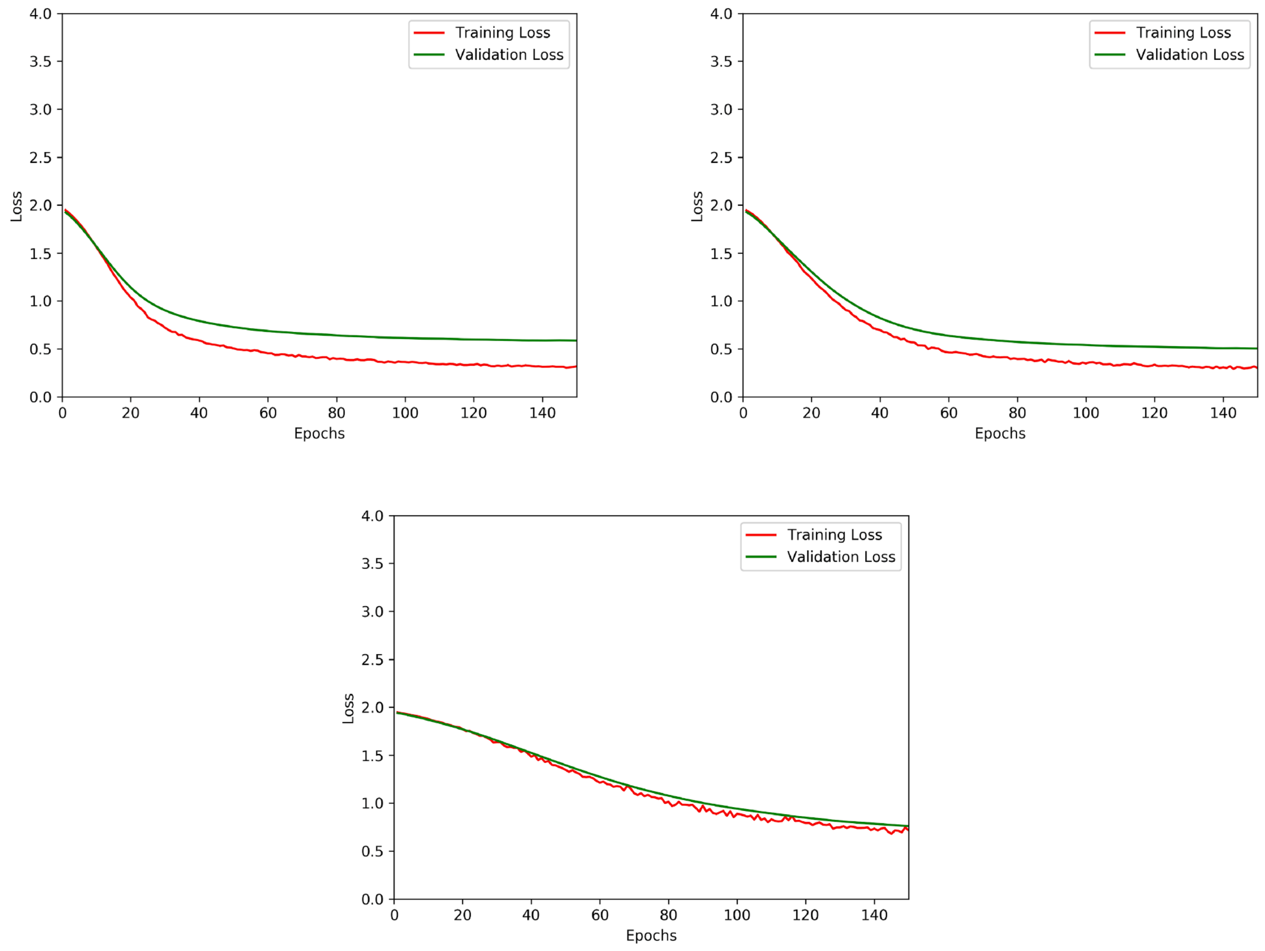


Fig. 6. (a) Cora loss convergence with structural features and no attention (b) Cora loss convergence with node features and no attention (c) Cora loss convergence with structural features, node features and no attention.

Finally, we consider the ogb-proteins network, which unlike the earlier networks, is a biological network expressing the relational information between functional protein components. Additionally, we note that this protein network is denser in comparison to the other ogb networks under study. Due to the density of the network, we were unable to fit any attention models in the GPU. We report our results in Table 6. Since this is a multi-class, multi-label classification, we report the Receiver Operating Characteristics-Area Under the Curve (ROC-AUC) score in percentage. We observe the KP-GCN model which does not consider any node features outperforms the state of the art methods. Further, we note that the classification performance of all methods improved in the presence of an identity matrix (absence of node features), which reveals the importance of self-attention as features for this network. Due to the nature/form in which the ogb-proteins network is available in the official ogb database, we were unable to convert it to a form suitable for ingestion by the official version of the GDC. As a result, we did not include the results pertaining to GDC-GCN and GDC-JK Net for this particular dataset. Further, we find that with exception to the following variant of KP-GCN ($[\mathbf{I}, \Pi^{*}\widehat{\mathbf{D}}^{-1}]$), using higher-order structural information deteriorated the performance of GCN and JK Net by using our approach. This highlights the presence of valuable information from self-attention as features as opposed to node features in the ogb-proteins network. Additionally, a model such as GCN with no residual connections like JK Net is observed to perform well in the presence of higher-order features and self-attention as opposed to just self-attention or higher-order structural features. Hence, the proteins network can be concluded to have significant information in the form of self-attention and higher-order information that can efficiently be learned using less complex models such as GCN.

Based on our experiments, the ideal configuration for achieving the best classification performance involves using both the structural and node attribute features to learn meaningful representations. However, from our experiments, we find that the configuration is dependent on the network. The kernel propagation layer essentially captures higher-order neighborhood information along with the local structure without the need for any attention mechanism even when attention models don't fit in memory. Since our method proposes a pre-processing step applied to GCN, GAT and JK Net, it exhibits the same memory and time complexity as these models with the addition of the time complexity required to compute the *t*-hop random walk matrix. Hence, by combining structural information in the form of higher powers of random walk matrix and positional features such as node features (You et al., 2019), our kernel

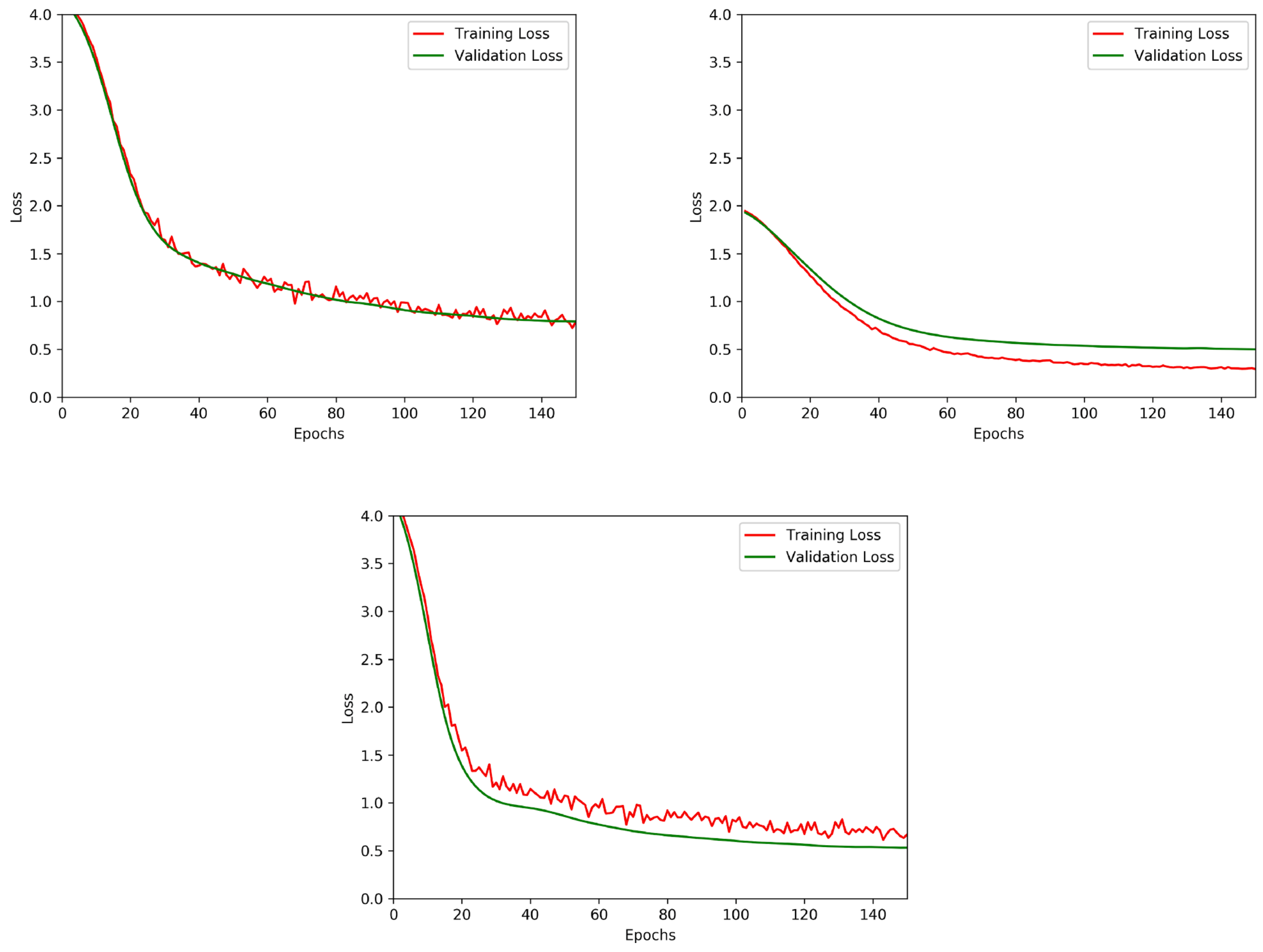


Fig. 7. (a) Cora loss convergence with structural features and attention (b) Cora loss convergence with node features and attention (c) Cora loss convergence with structural features, node features and attention.

Table 3
Classification accuracies on small-scale real world networks ([.,.] denotes feature concatenation).

Method	Data used	Cora	Citeseer	Pubmed
Node2vec (Grover and Leskovec, 2016)	**A**	68.66±1.83%	47.98±1.75%	72.36±0.95%
GCN (Kipf and Welling, 2016)	**A, I, Y**	84.10±0.34%	68.75±0.67%	79.70±1.33%
GCN (Kipf and Welling, 2016)	**A, X, Y**	85.58±0.48%	71.43±0.47%	84.00±1.10%
GAT (Veličković et al., 2017)	**A, I, Y**	82.40±0.62%	71.25±0.43%	83.90±0.70%
GAT (Veličković et al., 2017)	**A, X, Y**	83.34±0.47%	72.71±0.52%	83.90±0.70%
JK Net (Xu et al., 2018)	**A, I, Y**	81.53±0.37%	70.34±0.29%	84.40±0.40%
JK Net (Xu et al., 2018)	**A, X, Y**	82.60±0.43%	71.70±0.54%	84.50±0.80%
APPNP (Klicpera et al., 2018)	**A, I, Y**	84.93±0.29%	74.84±0.46%	83.37±0.32%
APPNP (Klicpera et al., 2018)	**A, X, Y**	85.79±0.27%	75.97±0.46%	84.10±0.72%
GDC-GCN (Klicpera et al., 1335)	**A, X, Y**	84.70±0.30%	71.25±0.27%	82.60±0.66%
KP-GCN	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	85.60±0.53%	72.75±0.90%	82.20±1.10%
AKP-GCN	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	84.60±0.60%	73.75±0.70%	81.10±0.70%
AKP-GCN	**A, X, Y**	86.00±0.60%	**76.00±1.00%**	84.60±1.10%
KP-GCN	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	87.00±0.80%	74.10±0.90%	**85.10±0.80%**
AKP-GCN	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	**87.80±0.80%**	**76.00±0.90%**	84.50±0.60%

propagating layer is also able to address the problem of sub-structure of isomorphism.

5. Conclusions

In this work, we show that computing node independent structural features as input features for each node in the network enables existing models to perform on par with or better than approaches which consider only node features by following a first-order spectral propagation scheme. Additionally, if such node features are unavailable, higher-order structural features can be used for node classification. Node features have shown to improve node classification because of the additional information it provides to differentiate between nodes which have similar sub-structure node arrangements. Hence node features can be used concurrently along with structural features. This work motivates research in the direction of deriving node independent structural features which can enable encoding higher-order neighborhood information alongside node features which complements structural information. Future work will explore independence between structural and node attribute features so that we understand the influence of topology and node features on a given network, to then inform the appropriate choice of network configuration for learning node representations from a diverse range of real-world networks.

Table 4
Classification accuracies ([.,.] denotes feature concatenation).

Method	Data used	ogbn-arxiv (Validation)	ogbn-arxiv (Test)
Node2vec (Grover and Leskovec, 2016)	**A**	69.23±0.13%	66.71±0.14%
GCN (Kipf and Welling, 2016)	**A, I, Y**	68.42±0.34%	66.88±0.89%
GCN (Kipf and Welling, 2016)	**A, X, Y**	71.22±0.03%	70.56±0.28%
GAT (Veličković et al., 2017)	**A, I, Y**	68.97±0.41%	67.62±0.74%
GAT (Veličković et al., 2017)	**A, X, Y**	71.71±0.04%	70.69±0.48%
JK Net (Xu et al., 2018)	**A, I, Y**	64.49±0.26%	61.35±0.62%
JK Net (Xu et al., 2018)	**A, X, Y**	71.29±0.09%	69.67±0.46%
APPNP (Klicpera et al., 2018)	**A, I, Y**	70.52±0.29%	69.14±0.43%
APPNP (Klicpera et al., 2018)	**A, X, Y**	71.39±0.19%	70.09±0.32%
GDC-GCN (Klicpera et al., 1335)	**T, X, Y**	69.44±0.01%	65.93±0.15%
GDC-JK Net (Klicpera et al., 1335)	**T, X, Y**	68.57±0.27%	65.21±0.45%
KP-GCN	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	69.98±0.12%	68.69±0.26%
KP-GAT	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	70.14±0.04%	68.82±0.17%
KP-JK Net	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	67.81±0.09%	65.78±0.04%
KP-GCN	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	71.55±0.30%	70.63±0.08%
KP-GAT	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	**72.08±0.11%**	**71.17±0.19%**
KP-JK Net	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	71.82±0.22%	70.20±0.21%

Table 5
Classification accuracies ([.,.] denotes feature concatenation).

Method	Data Used	ogbn-products (Validation)	ogbn-products (Test)
Node2vec (Grover and Leskovec, 2016)	**A**	86.32±0.06%	68.49±0.10%
GCN (Kipf and Welling, 2016)	**A, I, Y**	90.97±0.07%	71.20±0.55%
GCN (Kipf and Welling, 2016)	**A, X, Y**	88.06±0.12%	68.86±0.26%
JK Net (Xu et al., 2018)	**A, I, Y**	87.50±0.06%	56.78±1.06%
JK Net (Xu et al., 2018)	**A, X, Y**	88.70±0.08%	69.57±0.06%
APPNP (Klicpera et al., 2018)	**A, I, Y**	91.69±0.05%	73.93±0.33%
APPNP (Klicpera et al., 2018)	**A, X, Y**	89.45±0.03%	73.05±0.07%
GDC-GCN (Klicpera et al., 1335)	**A, X, Y**	87.81±0.06%	66.06±0.08%
GDC-JK Net (Klicpera et al., 1335)	**A, X, Y**	88.83±0.09%	67.17±0.03%
KP-GCN	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	91.85±0.07%	74.16±0.41%
KP-GCN	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	91.85±0.12%	74.04±0.10%
KP-JK Net	$\mathbf{A}, \Pi^*\hat{\mathbf{D}}^{-1}, \mathbf{Y}$	91.23±0.05%	70.03±0.12%
KP-JK Net	$\mathbf{A}, [\mathbf{X}, \Pi^*\hat{\mathbf{D}}^{-1}], \mathbf{Y}$	**92.34±0.05%**	**74.80±0.66%**

Table 6
ROC-AUC score ([.,.] denotes feature concatenation).

Method	Data used	ogbn-proteins (Validation)	ogbn-proteins (Test)
Node2vec (Grover and Leskovec, 2016)	**A**	66.34±0.56%	64.11±0.67%
GCN (Kipf and Welling, 2016)	**A, I, Y**	80.00±0.16%	76.71±1.21%
GCN (Kipf and Welling, 2016)	**A, X, Y**	75.77±0.26%	69.05±0.47%
JK Net (Xu et al., 2018)	**A, I, Y**	74.91±0.26%	72.83±0.40%
JK Net (Xu et al., 2018)	**A, X, Y**	77.57±0.24%	73.84±0.59%
APPNP (Klicpera et al., 2018)	**A, I, Y**	80.21±0.52%	76.21±0.95%
APPNP (Klicpera et al., 2018)	**A, X, Y**	70.99±0.19%	65.43±0.25%
KP-GCN	$\mathbf{A}, \Pi^*\widehat{\mathbf{D}}^{-1}, \mathbf{Y}$	72.82±0.38%	65.91±0.90%
KP-GCN	$\mathbf{A}, [\mathbf{I}, \Pi^*\widehat{\mathbf{D}}^{-1}], \mathbf{Y}$	**81.14±0.27%**	**77.97±0.28%**
KP-GCN	$\mathbf{A}, [\mathbf{X}, \Pi^*\widehat{\mathbf{D}}^{-1}], \mathbf{Y}$	74.77±0.64%	68.00±0.92%
KP-JK Net	$\mathbf{A}, \Pi^*\widehat{\mathbf{D}}^{-1}, \mathbf{Y}$	67.41±3.07%	67.07±3.18%
KP-JK Net	$\mathbf{A}, [\mathbf{I}, \Pi^*\widehat{\mathbf{D}}^{-1}], \mathbf{Y}$	72.43±0.76%	70.50±1.46%
KP-JK Net	$\mathbf{A}, [\mathbf{X}, \Pi^*\widehat{\mathbf{D}}^{-1}], \mathbf{Y}$	66.46±0.17%	62.63±2.29%

Funding

This research was funded in part by the Air Force Office of Scientific Research (AFOSR) grant FA9550-18-1-0108. Any opinions, findings, or conclusions found in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

CRediT authorship contribution statement

Sakthi Kumar Arul Prakash: Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Conrad S. Tucker:** Conceptualization, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of convergence for the kernel propagation technique

Lemma 1. A mapping $\mathscr{M}: \Pi \longrightarrow \Pi$ is a contraction mapping.

$$\|\mathscr{M}(\widetilde{\Pi}) - \mathscr{M}(\Pi)\|_\infty \leqslant \gamma \|\widetilde{\Pi} - \Pi\|_\infty \tag{9}$$

where, $\mathscr{M}(\widetilde{\Pi}) = \mathbf{I}_N + \gamma\widehat{\mathbf{P}}\widetilde{\Pi}$ and $\mathscr{M}(\Pi) = \mathbf{I}_N + \gamma\widehat{\mathbf{P}}\Pi$

Proof.

$$\|\mathscr{M}(\widetilde{\Pi}) - \mathscr{M}(\Pi)\|_\infty = \|\mathbf{I}_N + \gamma\widehat{\mathbf{P}}\widetilde{\Pi} - \mathbf{I}_N - \gamma\widehat{\mathbf{P}}\Pi\|_\infty = \|\gamma\widehat{\mathbf{P}}\left(\widetilde{\Pi} - \Pi\right)\|_\infty \leqslant \gamma\|\widehat{\mathbf{P}}\|_\infty\|\widetilde{\Pi} - \Pi\|_\infty \tag{10}$$

Since we know that a transition probability matrix is a right stochastic matrix, $\left\|\widehat{\mathbf{P}}\right\|_\infty = \max_i \sum_j \widehat{\mathbf{P}}(i,j) = 1$. Hence, Eq. (4) simplifies to,

$$\|\mathscr{M}(\widetilde{\Pi}) - \mathscr{M}(\Pi)\|_\infty \leqslant \gamma \|\widetilde{\Pi} - \Pi\|_\infty \tag{11}$$

Appendix B. Additional experimental details

All experiments were conducted on a desktop with the following configuration:

- Operating System: Ubuntu 18.04.5 LTS
- CPU: Intel Xeon(R) CPU E5-2698 v4
- GPU: Tesla V100-DGXS-32 GB
- Software: Python 3.7.5, Pytorch 1.4.0.

References

Girvan, M. & Newman, M. E. J. (2002). Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99 (12), 7821–7826. ISSN 0027-8424.

Rosvall, M. & Bergstrom, C.T. (2008). Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences 105 (4), 1118–1123. ISSN 0027-8424.

Lu, Q. & Getoor, L. (2003). Link-based classification. In Proceedings of the 20th international conference on machine learning (ICML-03) (pp. 496–503).

Perozzi, B., Al-Rfou, R. & Skiena, S. (2014). 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. ISBN 145032956X.

Grover, A. & Leskovec, J. (2016). 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. ISBN 1450342329.

Kipf, T. N. & Welling, M., 2016. Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907.

Li, B. & Pi, D. (2019). Learning deep neural networks for node classification. Expert Systems with Applications 137, 324–334. ISSN 0957-4174.

Bhagat, S., Cormode, G., & Muthukrishnan, S. (2011). Node classification in social networks. In *Social network data analytics* (pp. 115–148). Springer.

Gärtner, T., Horváth, T. & Wrobel, S. (2010). Graph kernels.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. & Philip, S. Y., 2020. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems. ISSN 2162–237X.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. & Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.

Hamilton, W., Ying, Z. & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in neural information processing systems (pp. 1024–1034).

Xu, K., Hu, W., Leskovec, J. & Jegelka, S. (2018). How powerful are graph neural networks? In International conference on learning representations.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. -I. & Jegelka, S., 2018. Representation learning on graphs with jumping knowledge networks. arXiv preprint arXiv:1806.03536.

Klicpera, J., Bojchevski, A. & Günnemann, S., 2018. Predict then propagate: Graph neural networks meet personalized pagerank, arXiv preprint arXiv:1810.05997.

Abu-El-Haija, S., Alipourfard, N., Harutyunyan, H., Kapoor, A. & Perozzi, B. (2018). A higher-order graph convolutional layer. In Proceedings of the 32nd conference on neural information processing systems (NIPS 2018). NIPS.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y. & Hjelm, R. D., 2018. Deep graph infomax. arXiv preprint arXiv:1809.10341.

Bianconi, G., Pin, P. & Marsili, M. (2009). Assessing the relevance of node features for network structure. Proceedings of the National Academy of Sciences 106 (28), 11433–11438. ISSN 0027-8424.

Klicpera, J., Weißenberger, S. & Günnemann, S. (2019). Diffusion improves graph learning. In Advances in neural information processing systems (pp. 13354–13366).

Belkin, M. & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In Advances in neural information processing systems (pp. 585–591).

Tang, L. & Liu, H. (2011). Leveraging social media networks for classification. Data Mining and Knowledge Discovery 23 (3), 447–478, ISSN 1384-5810.

Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., Koutra, D., Faloutsos, C. & Li, L. (2012). Rolx: Structural role extraction & mining in large graphs. In Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1231–1239). ACM. ISBN 1450314627.

Ribeiro, L. F. R., Saverese, P. H. P. & Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 385–394). ACM. ISBN 1450348874.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K. & Tang, J. (2018). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In Proceedings of the eleventh ACM international conference on web search and data mining (pp. 459–467). ACM. ISBN 1450355811.

Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E, 69*(2), 26113.

Yang, J. & Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems 42 (1), 181–213. ISSN 0219-1377.

Wang, X., Jin, D., Cao, X., Yang, L. & Zhang, W. (2016). Semantic community identification in large attribute networks. In Thirtieth AAAI conference on artificial intelligence.

He, D., Feng, Z., Jin, D., Wang, X. & Zhang, W. (2017). Joint identification of network communities and semantics via integrative modeling of network topologies and node contents. In Thirty-first AAAI conference on artificial intelligence.

Li, Y., Sha, C., Huang, X. & Zhang, Y. (2018). Community detection in attributed graphs: an embedding approach. In Thirty-second AAAI conference on artificial intelligence.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111–3119).

Bahdanau, D., Cho, K. & Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Luong, M. -T., Pham, H. & Manning, C. D., 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).

Mnih, V., Heess, N. & Graves, A. (2014). Recurrent models of visual attention. In Advances in neural information processing systems (pp. 2204–2212).

Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K. & Koh, E. (2019). Attention models in graphs: A survey. ACM Transactions on Knowledge Discovery from Data (TKDD) 13 (6), 1–25. ISSN 1556-4681.

Li, Q., Han, Z. & Wu, X. -M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32). ISBN 2374-3468.

Chung, F. (2007). The heat kernel as the pagerank of a graph. Proceedings of the National Academy of Sciences 104 (50), 19735–19740. ISSN 0027-8424.

Donnat, C., Zitnik, M., Hallac, D. & Leskovec, J., 2018. Spectral graph wavelets for structural role similarity in networks.

You, J., Ying, R. & Leskovec, J., 2019. Position-aware graph neural networks. arXiv preprint arXiv:1906.04817.

Smola, A. J., & Kondor, R. (2003). Kernels and regularization on graphs. In *Learning theory and kernel machines* (pp. 144–158). Springer.

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

Oono, K. & Suzuki, T., 2019. Graph neural networks exponentially lose expressive power for node classification. arXiv preprint arXiv:1905.10947.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M. & Leskovec, J.,2020. Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Kingma, D. P. & Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.