Graph Convolutional Networks with Motif-based Attention

John Boaz Lee
Worcester Polytechnic Institute
jtlee@wpi.edu

Ryan A. Rossi
Adobe Research
rrossi@adobe.com

Xiangnan Kong
Worcester Polytechnic Institute
xkong@wpi.edu

Sungchul Kim
Adobe Research
sukim@adobe.com

Eunyee Koh
Adobe Research
eunyee@adobe.com

Anup Rao
Adobe Research
anuprao@adobe.com

ABSTRACT

The success of deep convolutional neural networks in the domains of computer vision and speech recognition has led researchers to investigate generalizations of the said architecture to graph-structured data. A recently-proposed method called Graph Convolutional Networks has been able to achieve state-of-the-art results in the task of node classification. However, since the proposed method relies on localized first-order approximations of spectral graph convolutions, it is unable to capture higher-order interactions between nodes in the graph. In this work, we propose a motif-based graph attention model, called Motif Convolutional Networks, which generalizes past approaches by using weighted multi-hop motif adjacency matrices to capture higher-order neighborhoods. A novel attention mechanism is used to allow each individual node to select the most relevant neighborhood to apply its filter. We evaluate our approach on graphs from different domains (social networks and bioinformatics) with results showing that it is able to outperform a set of competitive baselines on the semi-supervised node classification task. Additional results demonstrate the usefulness of attention, showing that different higher-order neighborhoods are prioritized by different kinds of nodes.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Human-centered computing** → *Social network analysis*; • **Computing methodologies** → *Reinforcement learning*; *Semi-supervised learning settings.*

KEYWORDS

Graph attention; motifs; graph convolution; higher-order proximity; structural role; deep learning

ACM Reference Format:
John Boaz Lee, Ryan A. Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. 2019. Graph Convolutional Networks with Motif-based Attention. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357880

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357880

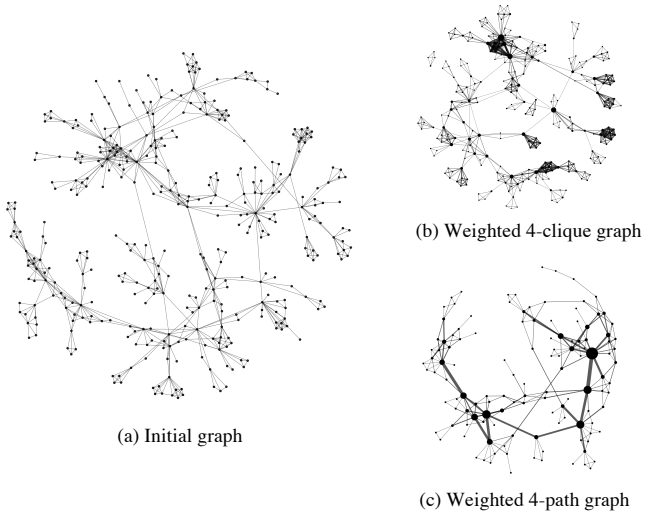

Figure 1: Node neighborhoods can differ significantly when we define adjacency based on higher-order structures or motifs. The size (weight) of nodes and edges in (b) and (c) correspond to the frequency of 4-node cliques and 4-node paths between nodes, respectively.

1 INTRODUCTION

In recent years, deep learning has made a significant impact on the field of computer vision. Various deep learning models have achieved state-of-the-art results on a number of vision-related benchmarks. In most cases, the preferred architecture is a Convolutional Neural Network (CNN). CNN-based models have been applied successfully to the tasks of image classification [21], image super-resolution [19], and video action recognition [10], among many others.

CNNs, however, are designed to work for data that can be represented as grids (*e.g.*, videos, images, or audio clips) and do not generalize well to graphs – which have more irregular structure. Due to this limitation, it cannot be applied directly to many real-world problems whose data come in the form of graphs – social networks [31] or collaboration/citation networks [24] in social network analysis, for instance.

A recent deep learning architecture, called Graph Convolutional Networks (GCN) [20] approximates the spectral convolution operation on graphs by defining a layer-wise propagation that is based on the one-hop neighborhood of nodes. The first-order filters used by GCNs were found to be useful and have allowed the model to beat many established baselines in the semi-supervised node classification task [20, 42].

However, in many cases, it has been shown that it may be beneficial to consider the higher-order structure in graphs [6, 25, 33, 35, 46]. In this work, we introduce a general class of graph convolution networks which utilize weighted multi-hop *motif* adjacency matrices [33] to capture higher-order neighborhoods in graphs. The weighted adjacency matrices are computed using various network motifs [33]. Fig. 1 shows an example of the node neighborhoods that are induced when we consider two different kinds of motifs, showing that the choice of motif can significantly alter the neighborhood structure of nodes.

Our proposed method, which we call Motif Convolutional Networks (MCN), uses a novel attention mechanism to allow each node to select the most relevant motif-induced neighborhood to integrate information from. Intuitively, this allows a node to select its one-hop neighborhood (as in classical GCN) when its immediate neighborhood contains enough information for the model to classify the node correctly but gives it the additional flexibility to select an alternative neighborhood (defined by higher-order structures) when the information in its immediate vicinity is too sparse and/or noisy for good classification.

The aforementioned attention mechanism is trained using reinforcement learning which rewards choices (*i.e*, actions) that consistently result in a correct classification.

The main contributions of this paper are summarized as follows:

- We propose a model that generalizes GCNs by introducing multiple weighted motif-induced adjacencies that capture various higher-order neighborhoods.
- We introduce a novel attention mechanism that allows the model to choose the best neighborhood for *each node* to integrate information from.
- We demonstrate the superiority of the proposed method by comparing against strong baselines on graphs from two different domains (social network and bioinformatics). In particular, we observed a gain of up to 5.6% over the next best method on graphs which did not exhibit homophily.
- We demonstrate the usefulness of attention by showing how different nodes prioritize different neighborhoods.

The rest of the paper is organized as follows. In Section 2, we provide a review of related literature. We then introduce the details of our proposed approach in Section 3. We discuss important experimental results in Section 4. Finally, we conclude the paper in the last section.

2 RELATED LITERATURE

Neural Networks for Graphs Initial attempts to adapt neural network models to work with graph-structured data started with recursive models that treated the data as directed acyclic graphs [11, 39]. Later on, more generalized models called Graph Neural Networks (GNN) were introduced to process arbitrary graph-structured data [13, 37].

Recently, with the rise of deep learning and the success of models such as recursive neural networks (RNN) [15, 48] for sequential data and CNNs for grid-shaped data, there has been a renewed interest in adapting some of these approaches to more general graph-structured data.

Some work introduced architectures tailored for more specific problem domains [9, 23] – like NeuralFPS [9] which is an end-to-end differentiable deep architecture which generalizes the well-known Weisfeiler-Lehman algorithm for molecular graphs – while others defined graph convolutions based on spectral graph theory [17]. Another group of methods attempt to substitute principled-yet-expensive graph convolutions using spectral approaches by using approximations of such. For instance, Defferrard et al. [8] used Chebyshev polynomials to approximate a smooth filter in the spectral domain while GCNs [20] further simplified the process by using simple first-order filters.

The model introduced by Kipf and Welling [20] has been shown to work well on a variety of graph-based tasks [20, 29, 45] and has spawned variants including [1, 42]. We introduce a generalization of GCN [20] in this work but we differ from past approaches in two main points: first, we use weighted motif-induced adjacencies to expand the possible kinds of node neighborhoods available to nodes, and secondly, we introduce a novel attention mechanism that allows each node to select the most relevant neighborhood to diffuse (or integrate) information.

Higher-order Structures with Network Motifs Network motifs [25] are fundamental building blocks of complex networks; investigation of such patterns usually lead to the discovery of crucial information about the structure and the function of many complex systems that are represented as graphs. Prill et al. [32] studied motifs in biological networks showing that the dynamical property of robustness to perturbations correlated highly to the appearance of certain motif patterns while Paranjape et al. [30] looked at motifs in temporal networks showing that graphs from different domains tend to exhibit very different organizational structures as evidenced by the type of motifs present.

Multiple work have demonstrated that it is useful to account for higher-order structures in different graph-based ML tasks [3, 28, 33, 46]. DeepGL [34] uses motifs as a basis to learn deep inductive relational functions that represent compositions of relational operators applied to a base graph function such as triangle counts. Rossi et al. [33] proposed the notion of *higher-order network embeddings* and demonstrated that one can learn better embeddings when various motif-based matrix formulations are considered.

Yang et al. [46] defined a hierarchical motif convolution for the task of subgraph identification for graph classification. Sankar et al. [36], on the other hand, proposes a graph convolution method designed primarily for heterogeneous graphs which utilizes motif-based connectivities. In a recent work, Morris et al. [28] has shown that standard GNN architectures such as GCN have the same expressiveness as the 1-dimensional WL graph isomorphism heuristic and hence both approaches suffer from similar shortcomings. They propose a generalization using higher-order structures for the task of graph classification.

Table 1: Table of notations.

Symbol	Definition
$\mathcal{G}$	Undirected graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$.
N	Number of nodes in $\mathcal{G}$, *i.e.*, $\lvert\mathcal{V}\rvert = N$.
$\mathcal{H}$	The set $\{H_1, \cdots, H_T\}$ of T network motifs (*i.e.*, induced subgraphs).
$\mathbf{A}_t$	$N \times N$ motif-induced adjacency matrix corresponding to motif H_t. $(\mathbf{A}_t)_{i,j}$ encodes the number of motifs of type H_t which contain $(i, j) \in \mathcal{E}$. When the subscript t is ommitted, this refers to the default edge-defined adjacency matrix.
$\tilde{\mathbf{A}}_t$	$N \times N$ adjacency matrix $\mathbf{A}_t$ with added self-loops.
$\tilde{\mathbf{D}}_t$	$N \times N$ diagonal degree matrix of $\tilde{\mathbf{A}}_t$.
D	Number of features or attributes per node.
$\mathbf{X}$	$N \times D$ attribute matrix.
$\mathbf{H}^{(l)}$	Node feature embedding inputted at layer l; $\mathbf{H}^{(1)} = \mathbf{X}$.
$\mathbf{W}^{(l)}$	Trainable embedding matrix at layer l.
$\mathcal{N}_i^{(\tilde{\mathbf{A}})}$	The set of neighbors of node i with respect to adjacency matrix $\tilde{\mathbf{A}}$, *i.e.*, $\{j \mid \tilde{A}_{i,j} \neq 0, \text{for } 1 \leq j \leq N\}$.
R_i	Reinforcement learning reward corresponding to training sample i. If we classify node i correctly then $R_i = 1$, otherwise $R_i = -1$.

Our work differs from previous approaches [28, 33, 34, 36, 46] in several key points. Specifically, in contrast to [28, 33, 34, 46], we propose a new class of higher-order network embedding methods which utilizes a novel motif-based attention for the task of semi-supervised node classification. The proposed method generalizes previous graph convolutional approaches [20, 42]. Also, unlike [36], we focus primarily on homogeneous graphs.

Attention Models Attention was popularized in the deep learning community as a way for models to attend to important parts of the data [4, 26]. The technique has been successfully adopted by models solving a variety of tasks. For instance, it was used by Mnih et al. [26] to take glimpses of relevant parts of an input image for image classification; on the other hand, Xu et al. [44] used attention to focus on task-relevant parts of an image for the image captioning task. Meanwhile, Bahdanau et al. [4] utilized attention for the task of machine translation by fixing the model attention on specific parts of the input when generating the corresponding output words.

There has also been a surge in interest at applying attention to deep learning models for graphs. The work of Velickovic et al. [42] used a node self-attention mechanism to allow each node to focus on features in its neighborhood that were more relevant while Lee et al. [22] used attention to guide a walk in the graph to learn an embedding for the graph. More specialized methods of graph attention models include [7, 14] with Choi et al. [7] using attention on a medical ontology graph for medical diagnosis and Han et al. [14] using attention on a knowledge graph for the task of entity link prediction. Our approach differs significantly, however, from previous approach in that we use attention to allow our model to select task relevant neighborhoods.

3 APPROACH

We begin this section by introducing the foundational layer that is used to construct arbitrarily deep motif convolutional networks. When certain constraints are imposed on our model's architecture, the model degenerates into a Graph Attention Network (GAT) [42] which, in turn, generalizes a GCN [20]. Because of this, we briefly introduce a few necessary concepts from [20, 42] before defining the actual neural architecture we employ – including the reinforcement learning strategy we use to train our attention mechanism.

3.1 Notation

We use upper-case bold letters to denote matrices, lower-case bold letters to represent vectors, and non-bold italicized letters for scalars. Frequently used notation is summarized in Table 1.

3.2 Graph Self-Attention Layer

A multi-layer GCN [20] is constructed using the following layer-wise propagation:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}). \tag{1}$$

Here, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the modified adjacency matrix of the input graph with added self-loops – $\mathbf{A}$ is the original adjacency matrix of the input undirected graph with N nodes while $\mathbf{I}_N$ represents an identity matrix of size N. The matrix $\tilde{\mathbf{D}}$, on the other hand, is the diagonal degree matrix of $\tilde{\mathbf{A}}$ (*i.e.*, $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$). Finally, $\mathbf{H}^{(l)}$ is the matrix of node features inputted to layer l while $\mathbf{W}^{(l)}$ is a trainable embedding matrix used to embed the given inputs (typically to a lower dimension) and σ is a non-linearity.

The term $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ in Eq. 1 produces a symmetric normalized matrix which updates each nodes representation via a weighted sum of the features in a node's one-hop neighborhood (the added self-loop allows the model to include the node's own features). Each link's strength (*i.e.*, weight) is normalized by considering the degrees of the corresponding pair of nodes. Formally, at each layer l, node i integrates neighboring features to obtain a new feature/embedding via:

$$\vec{\mathbf{h}}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i^{(\tilde{\mathbf{A}})}} \alpha_{i,j}\vec{\mathbf{h}}_j^{(l)}\mathbf{W}^{(l)}\right), \tag{2}$$

where $\vec{\mathbf{h}}_i^{(l)}$ is the feature vector of node i at layer l, with fixed weights $\alpha_{i,j} = \frac{1}{\sqrt{\deg(i)\deg(j)}}$, and $\mathcal{N}_i^{(\tilde{\mathbf{A}})}$ is the set of i's neighbors defined by the matrix $\tilde{\mathbf{A}}$ – which includes itself.

In GAT [42], Eq. 2 is modified with weights α that are differentiable or trainable and this can be viewed as follows,

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}[\vec{\mathbf{h}}_i\mathbf{W}\ \vec{\mathbf{h}}_j\mathbf{W}]\right)\right)}{\sum_{k \in \mathcal{N}_i^{(\tilde{\mathbf{A}})}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}[\vec{\mathbf{h}}_i\mathbf{W}\ \vec{\mathbf{h}}_k\mathbf{W}]\right)\right)}. \tag{3}$$

The attention vector $\mathbf{a}$ in Eq. 3 is a trainable weight vector that assigns importance to the different neighbors of i allowing the model to highlight particular neighboring node features that are more task-relevant.

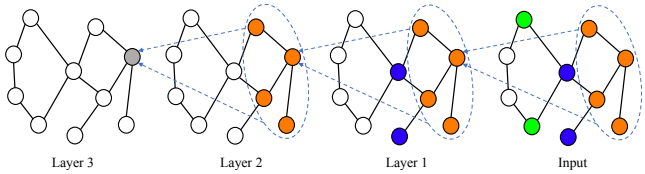

Figure 2: We can view a GCN [20, 42] as a message-passing algorithm. Each additional layer in a GCN allows the model to integrate information from a wider neighborhood. We illustrate this from the perspective of a target node (in gray). The target node integrates information from its one-hop neighbors (in orange) in layer 3. Previously, in layer 2, the orange nodes integrated information from their own one-hop neighborhood. Thus the target node also receives information from its two-hop neighbors (in blue). Similarly, in layer 1, the blue nodes integrated information from their immediate neighbors which results in the target node receiving information from its three-hop neighbors (in green). Image best viewed in color.

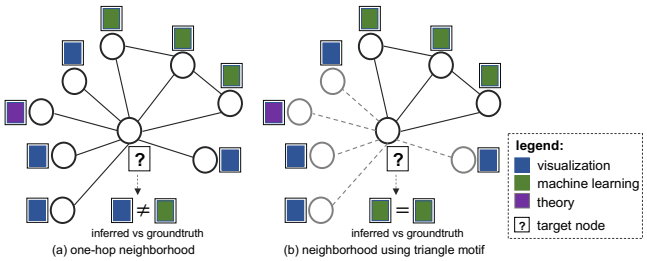

Figure 3: A researcher (target node) may have collaborated on various projects in visualization and theory. However, his main research focus is ML and he collaborates closely with lab members who also work among themselves. (a) If we simply use the target node's one-hop neighborhood, we may incorrectly infer his research area; however, (b) when we limit his neighborhood using the triangle motif, we reveal neighbors connected via stronger bonds giving us a better chance at inferring the correct research area. This observation is empirically shown in our experimental results. Illustration best viewed in color.

Using the formulation in Eq. 3 with Eqs. 1 and 2, we can now define multiple layers which can be stacked together to form a deep GCN (with self-attention) that is end-to-end differentiable. The initial input to the model can be set as $\mathbf{H}^{(1)} = \mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the initial node attribute matrix with D attributes. The final layer's weight matrix can also be set accordingly to output node embeddings at the desired output dimensions.

Figure 2 illustrates how an L-layer GCN (or GAT) enables a node to integrate information from its L-hop neighborhood. We see that this is done via repeated propagation via each nodes' one-hop neighborhood, layer by layer. Also, the size of the final neighborhood that information is propagated through is equivalent to the depth of the model.

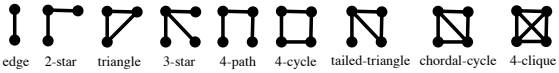

Figure 4: Network motifs or graphlets of sizes 2-4.

3.3 Convolutional Layer with Motif Attention

We observe that both GCN and GAT rely on the edge-defined one-hop neighborhood of nodes (*i.e.*, $\tilde{\mathbf{A}}$ in Eq. 1) to propagate information. However, it may not always be suitable to apply a single uniform definition of node neighborhood for all nodes. For instance, we show an example in Fig. 3 where a node can benefit from using a neighborhood defined using triangle motifs to keep only neighbors connected via a stronger bond which is a well-known concept from social theory allowing us to distinguish between weaker ties and strong ones via the triadic closure [12].

3.3.1 Weighted Motif-Induced Adjacencies. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes, $M = |\mathcal{E}|$ edges, as well as a set of T network motifs $\mathcal{H} = \{H_1, \cdots, H_T\}$, we can construct a set of T different motif-induced adjacency matrices $\mathcal{A} = \{\mathbf{A}_1, \cdots, \mathbf{A}_T\}$ where $\mathbf{A_t}$ is defined as follows:

$$(\mathbf{A}_t)_{i,j} = \#\text{ of motifs of type } H_t \text{ which contains both } i \text{ and } j.$$

In this paper, we use a loose definition for motifs and it can also mean induced subgraphs (*e.g.*, graphlets or orbits [2]). Motifs of sizes 2-4 are shown in Fig. 4. As shown in Fig. 1, neighborhoods defined by different motifs can vary significantly. Furthermore, the weights in a motif-induced adjacency $\mathbf{A}_t$ can also vary as motifs can appear in varying degrees of frequency between different pairs of nodes.

3.3.2 Motif Matrix Functions. Each of the calculated motif adjacencies $\mathbf{A}_t \in \mathcal{A}$ can now be potentially used to define motif-induced neighborhoods $\mathcal{N}_i^{(\mathbf{A}_t)}$ with respect to a node i. While Eq. 3 defines self-attention weights over a node's neighborhood, the initial weights in $\mathbf{A}_t$ can still be used as reasonable initial estimates of each neighbor's "importance."

Hence, we introduce a *motif-based matrix formulation* as a function $\Psi : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times N}$ over a motif adjacency $\mathbf{A}_t \in \mathcal{A}$ similar to [33]. Given a function Ψ, we can obtain *motif-based matrices* $\tilde{\mathbf{A}}_t = \Psi(\mathbf{A}_t)$, for $t = 1, \cdots, T$. Below, we summarize the different variants of Ψ that we chose to investigate.

• **Unweighted Motif Adjacency w/ Self-loops:** In the simplest case, we can construct $\tilde{\mathbf{A}}$ (here on, we omit the subscripts t for brevity) from $\mathbf{A}$ by simply ignoring the weights:

$$\tilde{\mathbf{A}}_{i,j} = \begin{cases} 1 & i = j \\ 1 & \mathbf{A}_{i,j} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

But, as mentioned above, we lose the initial benefit of leveraging the weights in the motif-induced adjacency $\mathbf{A}$.

• **Weighted Motif Adjacency w/ Row-wise Max:** We can also choose to retain the weighted motif adjacency $\mathbf{A}$ without modification save for added row-wise maximum self-loops. This is defined as follows:

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{M}, \tag{5}$$

where $\mathbf{M}$ is a diagonal square matrix with $M_{i,i} = \max_{1 \leq j \leq N} A_{i,j}$. Intuitively, this allows us to assign an equal amount of importance

to a self-loop consistent with that given to each node's most important neighbor.

• **Motif Transition w/ Row-wise Max:** The random walk on the weighted graph with added row-wise maximum self-loops has transition probabilities $P_{i,j} = \frac{A_{i,j}}{(\sum_k A_{i,k})+(\max_{1\leq k\leq N} A_{i,k})}$. Our random walk motif transition matrix can thus be calculated by

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1}(\mathbf{A} + \mathbf{M}), \tag{6}$$

where, in this context, the matrix $\mathbf{D}$ is the diagonal square degree matrix of $\mathbf{A} + \mathbf{M}$ (*i.e.*, $D_{i,i} = (\sum_k A_{i,k}) + (\max_{1\leq k\leq N} A_{i,k})$) while $\mathbf{M}$ is defined as above. Here, $\tilde{A}_{i,j} = P_{i,j}$ or the transition probability from node i to j is proportional to the motif count between nodes i and j relative to the total motif count between i and all its neighbors.

• **Absolute Motif Laplacian:** The absolute Laplacian matrix can be constructed as follows:

$$\tilde{\mathbf{A}} = \mathbf{D} + \mathbf{A}. \tag{7}$$

Here, the matrix $\mathbf{D}$ is the degree matrix of $\mathbf{A}$. Note that because the self-loop is a sum of all the weights to a node's neighbors, the initial importance of the node itself can be disproportionately large.

• **Symmetric Normalized Matrix w/ Row-wise Max:** Finally, we calculate a symmetric normalized matrix (similar to the normalized Laplacian) via:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{M})\mathbf{D}^{-\frac{1}{2}}. \tag{8}$$

Here, based on the context, the matrix $\mathbf{D}$ is the diagonal degree matrix of $\mathbf{A} + \mathbf{M}$.

3.3.3 K-Step Motif Matrices. Given a step-size K, we further define K different k-step motif-based matrices for each of the T motifs which gives a total of $K \times T$ adjacency matrices. Formally, this is formulated as follows:

$$\tilde{\mathbf{A}}_t^{(k)} = \Psi(\mathbf{A}_t^k), \text{ for } k = 1, \cdots, K \text{ and } t = 1, \cdots, T \tag{9}$$

where

$$\Psi(\mathbf{A}_t^k) = \Psi(\underbrace{\mathbf{A}_t \cdots \mathbf{A}_t}_{k}) \tag{10}$$

When we set $K > 1$, we allow nodes to accumulate information from a wider neighborhood. For instance if we choose to use Eq. 4 (for Ψ) and use an edge as our motif, $\tilde{\mathbf{A}}^{(k)}$ (we omit the motif-type subscript here) then captures k-hop neighborhoods of each node. While, in theory, using $\tilde{\mathbf{A}}^{(k)}$ is equivalent to using a k-layer GCN or GAT model, extensive experiments by Abu-El-Haija et al. [1] have shown that GCNs don't necessarily benefit from a wider receptive field as a result of increased model depth. This may be for reasons similar as to why skip-connections are needed in deep architectures since the signal starts to degrade as the model gets deeper [16].

As another example, we set Ψ to Eq. 6. Now for an arbitrary motif, we see that $(\tilde{\mathbf{A}}^{(k)})_{i,j}$ encodes the probability of transitioning from node i to node j in k steps.

While the K-step motif-based adjacencies defined here share some similarity to that of Rossi et al. [33] we would like to point out that there is an important distinction with our formulation. In particular, since graph convolutions integrate a node's own features via a self-loop we needed to define reasonable weights for the self-loops in the weighted adjacencies (*i.e.*, the diagonal) so that a node's information is not "overpowered" by its neighbors' features.

3.3.4 Motif Matrix Selection via Attention. Given T different motifs and a step-size of K, we now have $K \times T$ motif matrices we could use with Eq. 1 to define layer-wise propagations. A simple approach would be to implement $K \times T$ independent GCN instances and concatenate the final node outputs before classification. However, this approach may have problems scaling when T and/or K is large which makes it unfeasible.

Instead, we propose to use an attention mechanism, at each layer, to allow each node to select *a single* most relevant neighborhood to integrate or accumulate information from. For a layer l, this can be defined by two functions $f_l : \mathbb{R}^{S_l} \to \mathbb{R}^T$ and $f'_l : \mathbb{R}^{S_l} \times \mathbb{R}^T \to \mathbb{R}^K$, where S_l is the dimension of the state-space for layer l. The functions' outputs are *softmaxed* to form probability distributions over $\{1, \cdots, T\}$ and $\{1, \cdots, K\}$, respectively. Essentially, what this means is that given a node i's state, the functions recommend the most relevant motif t and step size k for node i to integrate information from.

Specifically, we define the state matrix encoding node states at layer l as a concatenation of two matrices:

$$\mathbf{S}_l = \left[\Psi(\mathbf{A})\mathbf{H}^{(l)}\mathbf{W}^{(l)} \quad \mathbf{C}\right], \tag{11}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{N\times D_l}$ is the weight matrix that embeds the inputs to dimension D_l, $\Psi(\mathbf{A})\mathbf{H}^{(l)}\mathbf{W}^{(l)}$ is the matrix containing local information obtained by doing a weighted sum of the features in the simple one-hop neighborhood for each node (from the original adjacency $\mathbf{A}$), and $\mathbf{C} \in \mathbb{R}^{N\times C}$ is a motif count matrix that gives us basic local structural information about each node by counting the number of C different motifs that each node belongs to. We note here that $\mathbf{C}$ is *not* appended to the node attribute matrix $\mathbf{X}$ and is not used for prediction. Its only purpose is to capture the local structural information of each node. $\mathbf{C}$ is computed once.

Let us consider an arbitrary layer. Recall that f (for brevity, we omit subscripts l) produces a probability vector specifying the importance of the various motifs, let $\vec{\mathbf{f}}_i = f(\vec{\mathbf{s}}_i)$ be the motif probabilities for node i. Similarly, let $\vec{\mathbf{f}}'_i = f'(\vec{\mathbf{s}}_i, \vec{\mathbf{f}}_i)$ be the probability vector recommending the step size. Now let t_i be the index of the largest value in $\vec{\mathbf{f}}_i$ and similarly, let k_i be the index of the largest value in $\vec{\mathbf{f}}'_i$. In other words, t_i is the recommended motif for i while k_i is the recommended step-size. Attention can now be used to define an $N \times N$ propagation matrix as follows:

$$\hat{\mathbf{A}} = \begin{bmatrix} \left(\tilde{\mathbf{A}}_{t_1}^{(k_1)}\right)_{1,:} \\ \vdots \\ \left(\tilde{\mathbf{A}}_{t_N}^{(k_N)}\right)_{N,:} \end{bmatrix}. \tag{12}$$

This layer-specific matrix $\hat{\mathbf{A}}$ can now be plugged into Eq. 1 to replace $\tilde{\mathbf{A}}$. What this does is it gives each node the flexibility to select the most appropriate motif t and step-size k to integrate information from.

3.3.5 Training the Attention Mechanism. Given a labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \ell)$ with N nodes and a labeling function $\ell : \mathcal{V} \to \mathcal{L}$

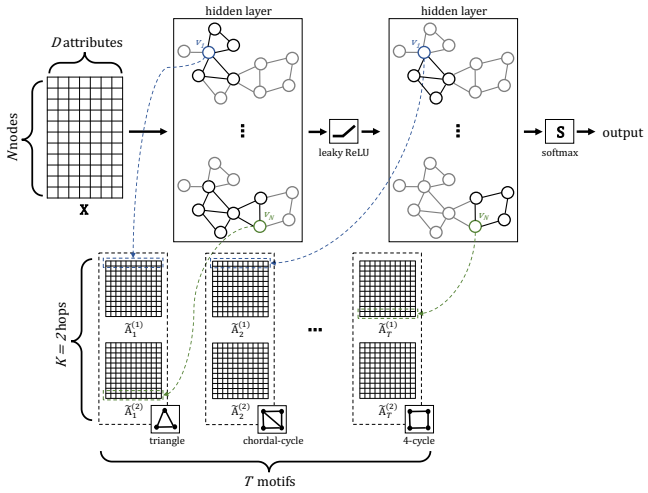

Figure 5: An example of a 2-layer MCN with $N = 11$ nodes, step-size $K = 2$, and T motifs. Attention allows each node to select a different motif-induced neighborhood to accumulate information from for each layer. For instance, in layer 1, the node v_N considers neighbors (up to 2-hops) that share a stronger bond (in this case, triangles) with it.

which maps each node to one of J class labels in $\mathcal{J} = \{1, \cdots, J\}$, our goal is to train a classifier that can predict the label of all the nodes. Given a subset $\mathcal{T} \subset \mathcal{V}$, or the training set of nodes, we can train an L-layer MCN (the classifier) using standard cross-entropy loss as follows:

$$\mathcal{L}_C = -\sum_{v \in \mathcal{T}} \sum_{j=1}^{J} Y_{vj} \log \pi(H_{i,j}^{(L+1)}), \tag{13}$$

where Y_{vj} is a binary value indicating node v's true label (*i.e.*, $Y_{vj} = 1$ if $\ell(v) = j$, zero otherwise), and $\mathbf{H}^{(L+1)} \in \mathbb{R}^{N \times L}$ is the *softmaxed* output of the MCN's last layer.

While Eq. 13 is sufficient for training the MCN to classify inputs it does not tell us how we can train the attention mechanism that selects the best motif and step-size for each node at each layer. We define a second loss function based on the REINFORCE rule:

$$\begin{aligned}\mathcal{L}_A = -\Bigg[& \sum_{n_L \in \mathcal{T}} R_v \left[\log \pi \left(\left(\vec{\mathbf{f}}_{n_L}^{(L)} \right)_{t_{n_L}^{(L)}} \right) + \log \pi \left(\left(\vec{\mathbf{f}}_{n_L}^{(L)} \right)_{k_{n_L}^{(L)}} \right) \right] \\ & + \sum_{n_L \in \mathcal{T}} \sum_{n_{L-1} \in \mathcal{N}_{n_L}^{(\hat{\mathbf{A}}^{(L)})}} R_v \left[\log \pi \left(\left(\vec{\mathbf{f}}_{n_{L-1}}^{(L-1)} \right)_{t_{n_{L-1}}^{(L-1)}} \right) + \log \pi \left(\left(\vec{\mathbf{f}}_{n_{L-1}}^{(L-1)} \right)_{k_{n_{L-1}}^{(L-1)}} \right) \right] \\ & + \cdots + \sum_{n_L \in \mathcal{T}} \cdots \sum_{n_1 \in \mathcal{N}_{n_2}^{(\hat{\mathbf{A}}^{(2)})}} R_v \left[\log \pi \left(\left(\vec{\mathbf{f}}_{n_1}^{(1)} \right)_{t_{n_1}^{(1)}} \right) + \log \pi \left(\left(\vec{\mathbf{f}}_{n_1}^{(1)} \right)_{k_{n_1}^{(1)}} \right) \right] \Bigg] \end{aligned} \tag{14}$$

Here, R_v is the reward we give to the system ($R_v = 1$ if we classify v correctly, $R_v = -1$ otherwise). The intuition here is this: at the last layer we reward the actions of the classified nodes; we then go to the previous layer (if there is one) and reward the actions of the neighbors of the classified nodes since their actions affect the outcome, we continue this process until we reach the first layer. Please refer to [26] for a more detailed explanation of the REINFORCE rule for reinforcement learning.

Table 2: Space of methods expressed by MCN. GCN and GAT are shown below to be special cases of MCN.

Method	Motif	Adj.	K	Self-attention	Motif-attention
GCN	edge	Eq. 4	$K = 1$	no	no
GAT	edge	Eq. 4	$K = 1$	yes	no
MCN-*	any	Eqs. 4-8	$K = \{1, \cdots\}$	yes	yes

Table 3: Dataset statistics. Value shown in brackets is the percentage of the nodes used for training.

	Cora	Citeseer	Pubmed
# of Nodes	2,708	3,327	19,717
# of Edges	5,429	4,732	44,338
# of Features/Node	1,433	3,703	500
# of Classes	7	6	3
# of Training Nodes	140 (5%)	120 (4%)	60 (<1%)

There are a few important things to point out. In practice, we use an ϵ-greedy strategy when selecting a motif and step-size during training. Specifically, we pick the action with highest probability most of the time but during $1 - \epsilon$ instances we select a random action. During testing, we choose the action with highest probability. Also, in practice, we use dropout to train the network as in GAT [42] which is a good regularization technique but also has the added advantage of being a way to sample the neighborhood during training to keep the receptive field from growing too large during training. Finally, to reduce model variance we can also include an advantage term (see Eq. 2 in [22], for instance). Our final loss can then be written as:

$$\mathcal{L} = \mathcal{L}_C + \mathcal{L}_A. \tag{15}$$

We show a simple (2-layer) example of the proposed MCN model in Fig. 5. As mentioned, MCN generalizes both GCN and GAT. We list settings of these methods in Table 2.

4 EXPERIMENTAL RESULTS

4.1 Semi-supervised node classification

We first compare our proposed approach against a set of strong baselines (including methods that are considered the current state-of-the-art) on three well-known graph benchmark datasets for semi-supervised node classification. We show that the proposed method is able to achieve state-of-the-art results on all compared datasets. The compared baselines are as follows:

- MLP: Standard fully-connected multi-layer perceptron. The model does not take into account graph structure and takes directly as input node features.
- LP [49]: Semi-supervised method based on Gaussian random fields which places both labeled and unlabeled samples on a weighted graph with weights representing pair-wise similarity.
- ICA [24]: A structured logistic regression model which leverages links between objects.
- ManiReg [5]: A framework that can be used for semi-supervised classification which uses a manifold-based regularization.

Table 4: Summary of experimental results: "average accuracy ± SD (rank)". The "Avg. Rank" column shows the average rank of each method. The lower the average rank, the better the overall performance of the method.

Method	Dataset			Avg. Rank
	Cora	Citeseer	Pubmed	
DeepWalk (Perozzi et al. [31])	67.2% (9)	43.2% (11)	65.3% (11)	10.3
MLP	55.1% (12)	46.5% (9)	71.4% (9)	10.0
LP (Zhu et al. [49])	68.0% (8)	45.3% (10)	63.0% (12)	10.0
ManiReg ([5])	59.5% (10)	60.1% (7)	70.7% (10)	9.0
SemiEmb (Weston et al. [43])	59.0% (11)	59.6% (8)	71.7% (8)	9.0
ICA (Lu and Getoor [24])	75.1% (7)	69.1% (5)	73.9% (7)	6.3
Planetoid (Yang et al. [47])	75.7% (6)	64.7% (6)	77.2% (5)	5.7
Chebyshev (Defferrard et al. [8])	81.2% (5)	69.8% (4)	74.4% (6)	5.0
MoNet (Monti et al. [27])	81.7% (3)	–	78.8% (4)	3.5
GCN (Kipf and Welling [20])	81.5% (4)	70.3% (3)	79.0% (2)	3.0
GAT (Velickovic et al. [42])	83.0 ± 0.7% (2)	72.5 ± 0.7% (2)	79.0 ± 0.3% (2)	2.0
MCN (this paper)	83.5 ± 0.4% (1)	73.3 ± 0.7% (1)	79.3 ± 0.3% (1)	1.0

- SemiEmb [43]: A model which integrates an unsupervised dimension reduction technique into a deep architecture to boost performance of semi-supervised learning.
- DeepWalk [31]: An unsupervised network embedding approach which uses the skip-gram algorithm to learn node embeddings that are similar for nodes that share a lot of links.
- Chebyshev [8]: A graph convolution approach which uses Chebyshev polynomials to approximate a smooth filter in the spectral domain.
- Planetoid [47]: A method which integrates graph embedding techniques into graph-based semi-supervised learning.
- MoNet [27]: A geometric deep learning approach that generalizes CNNs to graph-structured data.
- GCN [20]: A method which approximates spectral graph convolutions using first-order filters.
- GAT [42]: Generalization of GCNs with added node-level self-attention.
- MCN (this paper): Our proposed graph attention model with motif-based attention.

4.1.1 Datasets. We compare all baselines using three established benchmark datasets, these are: Cora, Citeseer, and Pubmed. Specifically, we use the pre-processed versions made available by Yang et al. [47]. The aforementioned graphs are undirected citation networks where nodes represent documents and edges denote citation; furthermore, a bag-of-words vector capturing word counts in each document serves as each node's feature. Each document is assigned a unique class label.

Following the procedure established in previous work, we use *only* 20 nodes per class for training [20, 42, 47]. Again, following previous work, we take 1,000 nodes per dataset for testing and utilize an additional 500 for validation [1, 20, 42]. We use the same train/test/validation splits as defined in [20, 42]. Statistics for the datasets is shown in Tab. 3.

4.1.2 Setup. For Cora and Citeseer, we used the same 2-layer model architecture as that of GAT consisting of 8 self-attention heads each with a total of 8 hidden nodes (for a total of 64 hidden nodes) in the first layer, followed by a single softmax layer for classification [42]. Similarly, we fixed early-stopping patience at 100 and ℓ_2-regularization at 0.0005. For Pubmed, we also used the same architecture as that of GAT (first layer remains the same but the output layer has 8 attention heads to deal with sparsity in the training data). Patience remains the same and similar to GAT, we use a strong ℓ_2-regularization at 0.001.

We further optimized all models by testing dropout values of $\{0.50, 0.55, 0.60, 0.65\}$, learning rates of $\{0.05, 0.005\}$, step-sizes $K \in \{1, 2, 3\}$, and motif adjacencies formed using combinations of the following motifs: *edge*, *2-star*, *triangle*, *3-star*, and *4-clique* (please refer to Fig. 4 for illustration of motifs).

Self-attention learns to prioritize neighboring features that are more relevant and the motif-based adjacencies derived from Ψ (Eqs. 4-8) can be viewed as reasonable initial estimates of self-attention. We select the initialization that yields the best result. Finally, we adopt an ϵ-greedy strategy ($\epsilon = 0.1$).

We note that for classification, our model uses *exactly* the same amount of information and the same number of model parameters as GAT [42] for fairness of comparison. The motif attention mechanism uses some additional trainable parameters to allow each node to select motifs but these parameters are separate from that of the classification network.

4.1.3 Comparison. For all three datasets, we report the classification accuracy averaged over 15 runs on random seeds (including standard deviation for methods that report these). A summary of the results is shown in Table 4. We see that our proposed method achieves superior performance against all compared baselines on all three benchmarks. On the Cora dataset, the best model used a learning rate of 0.005, dropout of 0.6, and both the edge and triangle motifs with step-size $K = 1$. For Citeseer, the learning rate was 0.05 and dropout was still 0.6 while the only motif used was the edge motif with step-size $K = 2$. However, the second best model for Citeseer – which had comparable performance – utilized the following motifs: edge, 2-star, and triangle. Finally, on Pubmed, the

Table 5: Micro-F1 scores of compared methods on DD.

Method	Dataset	
	DD-6	DD-7
GCN	11.9 ± 0.6%	12.4 ± 0.8%
GAT	11.8 ± 0.5%	11.8 ± 1.1%
MCN	**12.4 ± 0.5%**	**13.1 ± 0.9%**

best model used learning rate 0.05 and dropout of 0.5. Once again, the best motifs were the edge and triangle motifs on $K = 1$.

One interesting observation that can be made is the fact that the triangle motif is consistently used by the top models on all the datasets. This highlights an important advantage of MCN over past approaches (*e.g.*, GCN & GAT) which are not able to handle neighborhoods based on higher-order structures such as triangles. The results indicate that it can be beneficial to consider stronger bonds (friends that are friends themselves) when selecting a neighborhood.

Our experimental results show that we can improve model performance simply by relaxing the notion of node neighborhoods by allowing the model to choose attention-guided motif-based neighborhoods. We argue that the performance gain from this subtle but important change is significant especially since both MCN and GAT use an equal number of parameters *for classification*.

We also conducted some experiments on a random version of MCN which does not use attention to select motif-based neighborhoods. From our tests, we find that the method cannot outperform MCN with attention and the performance drops especially if there is a large number of motifs.

4.2 Comparison on Networks with Heterophily

The benchmark datasets (Cora, Citeseer, and Pubmed) that we initially tested our method on exhibited strong homophily where nodes that share the same labels tend to form densely connected communities. Under these circumstances, methods like GAT or GCN that use a first-order propagation rule will perform reasonably well. However, not all real-world graphs share this characteristic and in some cases the node labels are more spread out. In this latter case, there is reason to believe that neighborhoods constructed using different motifs – other than just edges and triangles – may be beneficial.

We test this hypothesis by comparing GAT and GCN against MCN on two graphs from the DD dataset [18]. Specifically, we chose two of the largest graphs in the dataset: DD-6 and DD-7 – with a total of 4, 152 and 1, 396 nodes, respectively. Both graphs had twenty different node labels with the labels being quite imbalanced.

We stick to the semi-supervised training regime, using only 15 nodes per class for training with the rest of the nodes split evenly between testing and validation. This makes the problem highly challenging since the graphs do not exhibit homophily. Since the nodes do not have any attributes, we use the WL algorithm (we initialize node attributes to a single value and run the algorithm for 3 iterations) to generate node attributes that capture each node's neighborhood structure [38] as in previous work [41].

For the three approaches (GCN, GAT, and MCN), we fix early-stop patience at 50 and use a two-layer architecture with 32 hidden nodes in the first layer followed by the softmax output. We optimized the hyperparameters by searching over learning rate in $\{0.05, 0.005\}$, ℓ_2 regularization in $\{0.01, 0.001, 0.0001, 0.00001\}$, and dropout at $\{0.2, 0.3, 0.4, 0.5, 0.6\}$. Furthermore, for MCN, we considered combinations of the following motifs {edge, 2-star, triangle, 4-path-edge, 3-star, 4-cycle, 4-clique} and considered K-steps from $1, \cdots, 4$. Since there are multiple classes and they are highly imbalanced, we report the Micro-F1 score averaged over 10 runs.

A summary of the results are shown in Table 5. These results demonstrate the effectiveness of MCN for realistic graphs that lack strong homophily. In particular, motif attention is shown to be extremely valuable as MCN achieves a 5.6% gain over the next best method for DD-7.

For DD-6, the best method utilized all motifs except for the 4-path-edge with $K = 1$ while in DD-7 the best approach used the edge, triangle, and 4-clique motifs with $K = 4$. In both cases, the model utilized multiple motifs.

Table 6: Statistics of large benchmark graphs. 'Edge %' denotes the ratio of the graph's edges versus the total number of edges in the largest dataset (LastFM).

Dataset	# of Nodes	# of Edges	Max Degree	Avg. Degree	Edge %
Cora	2,708	5,429	168	~4	<1.0%
Delicious	~536K	~1.4M	~3K	~5	31.1%
YouTube-Snap	~1.1M	~3M	~29K	~5	66.7%
LastFM	~1.2M	~4.5M	~5K	~7	100.0%

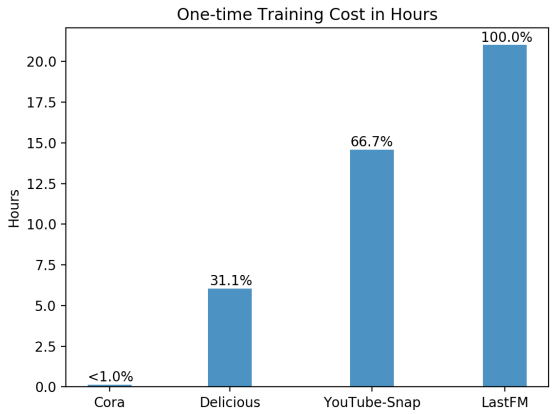

Figure 6: Runtime of proposed method on large real-world graphs. Percent values above the bars indicate the ratio of the dataset's edges compared to the number of edges in the largest dataset (LastFM).

4.3 Visualizing Motif Attention

We ran an instance of MCN ($K = 1$) on the Cora dataset with the following motifs: edge, 4-path, and triangle. Fig. 7 shows the nodes from two of the larger classes (class 3 and class 4) with each node colored by the motif that was selected by the attention mechanism.

Three important and interesting observations can be made here, we summarize them below.

- First, we find evidence of the model taking advantage of the flexibility provided by the attention mechanism to select a different motif-induced neighborhood for each node. We

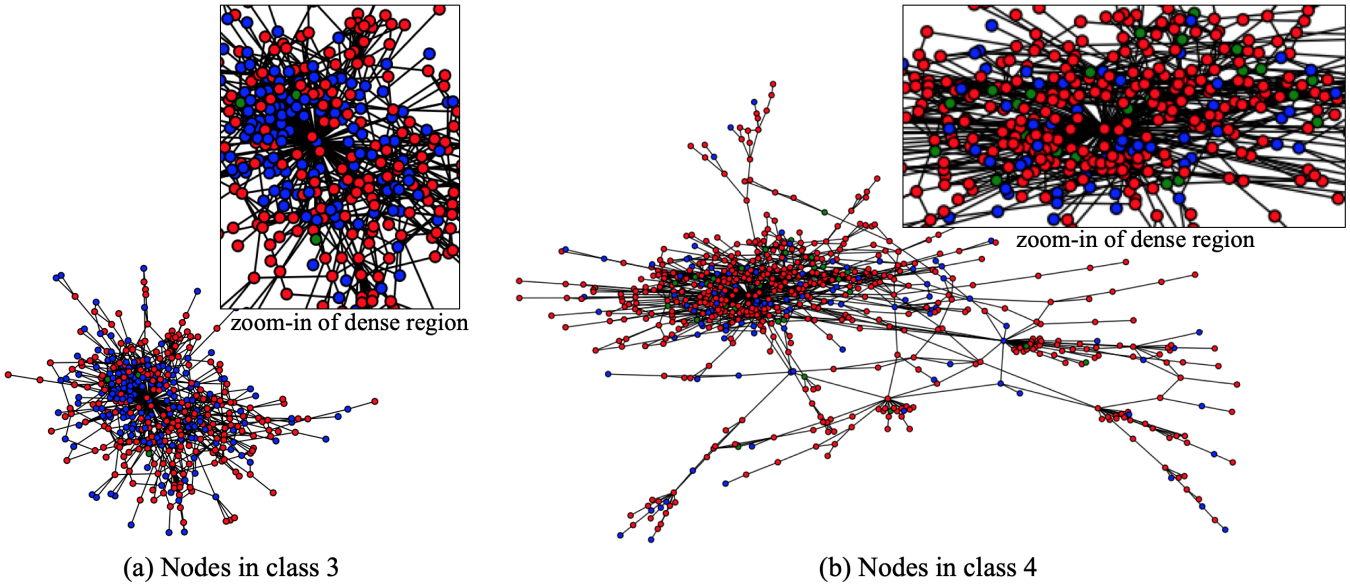

Figure 7: The largest connected components taken from the two induced subgraphs in Cora of nodes from (a) class 3 and (b) class 4, respectively. Nodes are colored to indicate the motif selected by the motif attention mechanism in the first layer. The motifs are: edge (blue), 4-path (red), and triangle (green). We observe that the nodes near the fringe of the cluster – particularly in (b) – tend to select the 4-path allowing them to aggregate information from a wider neighborhood. On the other hand, nodes that choose the triangle motif are fewer in number and can be found in the denser regions where it may be helpful to take advantage of stronger bonds. Image best viewed in color.

observe that *all three* types of motifs are selected and the model is not simply "defaulting" to a single type. Since our model can generalize to GAT [42], it can very well choose to just utilize the edge-based connections for every node if the other motif-based neighborhoods were not necessary.

- Second, we note that nodes that chose the triangle motif appear predominantly in denser parts of the cluster. This shows that it can be beneficial in these cases to consider the many strong bonds in the dense parts (especially if these nodes also share connections with nodes from other classes, *e.g.*, there is noise). For class 3, we observe 3 nodes selecting the neighborhood based on the triangle motif while more than 20 nodes chose the triangle motif for class 4.
- Lastly, we notice that nodes at the fringe of the cluster often prioritized the 4-path motif. This is quite intuitive since this allows the fringe nodes to aggregate information from a wider (4-hop) neighborhood which is useful since they are more separated from the other nodes in the same class.

4.4 Runtime on Large-scale Datasets

In the paper, we report semi-supervised classification results for smaller datasets as these are the standard graph benchmarks used by previous work [20, 42, 47] and also because these datasets have ground-truth node labels. However, the approach is fast and scalable for larger graph data. We demonstrate this in experiments on several large real-world social networks.

We benchmark a sparse implementation of our proposed method on three large real-world social networks: Delicious, Youtube-Snap, and LastFM[1]. For reference, we also include Cora. The statistics for these datasets are shown in the Tab. 6.

In our tests, we used the architecture of the model which performed the best in previous experiments. Specifically, we used a two-layer MCN with 8 self-attention heads (each with 8 hidden nodes) in the first layer and a softmax binary classification layer in the second layer. We tested the model with the following motifs: edge, triangle, and 4-clique. These were shown to give good performance in all our previous tests with $K = 1$ and weighted motif adjacencies. Finally, we used 5% of the total number of nodes for training and used an equal number for validation and testing. Since the graphs do not have corresponding node attributes, we randomly generated 50-dimensional node features for each node. Likewise we also assigned random class labels to the nodes.

We report the average one-time training runtime (over five runs) of our model when run for 400 epochs – which we have found in previous experiments to be sufficient in most cases for convergence. All experiments were performed on a MacBook Pro with 2.2 GHz Intel Core i7 processors and 16GB of RAM.

The plot in Fig. 6 shows the one-time training cost for the model on four large real-world datasets. Once the model is trained, the parameters can be loaded and prediction can be performed in $O(1)$ or constant time. We observe that training time does not exceed 21 hours for any of the datasets which is reasonable especially since the experiments were conducted on a standard work laptop. Also, the increase in runtime seems to be roughly linear with respect

[1]These are available at http://networkrepository.com

to the number of edges in the graph which is helpful since many real-world graphs are quite sparse [40].

5 CONCLUSION

In this work, we introduced a new class of higher-order network embedding methods which generalizes both GCN and GAT. The proposed model utilizes a novel motif-based attention for the task of semi-supervised node classification. Attention is used to allow different nodes to select the most task-relevant neighborhood to integrate information from.

Experiments on three citation (Cora, Citeseer, & Pubmed) and two bioinformatic (DD-6 & DD-7) benchmark graphs show the advantage of the proposed approach over previous work. We also show experimentally that different nodes do utilize attention to select different neighborhoods, indicating that it may be useful to consider various motif-defined neighborhoods. In particular, we found that neighborhoods defined by the triangle motif seemed to be especially useful. Finally, we benchmark a sparse implementation of MCN on several large real-world graphs and showed that the method can be run reasonably fast on large-scale networks.

ACKNOWLEDGEMENTS

This work is supported in part by National Science Foundation through grant IIS-1718310.

REFERENCES

[1] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. 2018. N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification. In *arXiv:1802.08888v1*.
[2] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, Nick Duffield, and Theodore L. Willke. 2017. Graphlet Decomposition: Framework, Algorithms, and Applications. *KAIS* 50, 3 (2017), 689–722.
[3] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2018. Learning Role-based Graph Embeddings. In *StarAI @ IJCAI*. 1–8.
[4] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*. 1–15.
[5] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR* 7 (2006), 2399–2434.
[6] Aldo G. Carranza, Ryan A. Rossi, Anup Rao, and Eunyee Koh. 2018. Higher-order Spectral Clustering for Heterogeneous Graphs. In *arXiv:1810.02959*. 1–15.
[7] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. 2017. GRAM: Graph-based Attention Model for Healthcare Representation Learning. In *KDD*. 787–795.
[8] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.
[9] David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gomez-Bombarelli, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *NIPS*. 2224–2232.
[10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *WSDM*. 601–610.
[11] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. 1998. A general framework for adaptive processing of data structures. *IEEE TNNLS* 9, 5 (1998), 768–786.
[12] Adrien Friggeri, Guillaume Chelius, and Eric Fleury. 2011. Triangles to Capture Social Cohesion. In *SocialCom/PASSAT*. 258–265.
[13] M. Gori, G. Monfardini, and F. Scarselli. 2005. A new model for learning in graph domains. In *IJCNN*. 729–734.
[14] Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural Knowledge Acquisition via Mutual Attention Between Knowledge Graph and Text. In *AAAI*. 1–8.
[15] Matthew Hausknecht and Peter Stone. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Symposium*. 1–9.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.
[17] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. In *arXiv:1506.05163v1*.
[18] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. 2016. Benchmark Data Sets for Graph Kernels. (2016). http://graphkernels.cs.tu-dortmund.de
[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *CVPR*. 1637–1645.
[20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. 1–14.
[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*. 1106–1114.
[22] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph Classification using Structural Attention. In *KDD*. 1666–1674.
[23] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*. 1–20.
[24] Qing Lu and Lise Getoor. 2003. Link-based classification. In *ICML*. 496–503.
[25] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science* 298, 5594 (2002), 824–827.
[26] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *NIPS*. 2204–2212.
[27] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. 2016. Deep Convolutional Networks on Graph-Structured Data. In *arXiv:1611.08402*.
[28] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *AAAI*. 1–16.
[29] Thien Huu Nguyen and Ralph Grishman. 2018. Graph Convolutional Networks with Argument-Aware Pooling for Event Detection. In *AAAI*. 5900–5907.
[30] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *WSDM*. 601–610.
[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[32] Robert J. Prill, Pablo A. Iglesias, and Andre Levchenko. 2005. Dynamic Properties of Network Motifs Contribute to Biological Network Organization. *PLoS Biology* 3, 11 (2005), 1881–1892.
[33] Ryan A. Rossi, Nesreen K. Ahmed, and Eunyee Koh. 2018. Higher-order Network Representation Learning. In *WWW*. 3–4.
[34] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *BigNet @ WWW*. 1–8.
[35] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Estimation of Graphlet Counts in Massive Networks. In *TNNLS*. 1–14.
[36] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. 2018. Motif-based Convolutional Neural Network on Graphs. In *arXiv:1711.05697v3*. 1–7.
[37] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE TNNLS* 20, 1 (2009), 61–80.
[38] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *JMLR* 12 (2011), 1–23.
[39] Alessandro Sperduti and Antonina Starita. 1997. Supervised neural networks for the classification of structures. *IEEE TNNLS* 8, 3 (1997), 714–735.
[40] Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. 2007. Less is More: Compact Matrix Decomposition for Large Sparse Graphs. In *SDM*. 366–377.
[41] Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. 2018. Learning Graph Representations with Recurrent Neural Network Autoencoders. In *Deep Learning Day @ KDD*. 1–8.
[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*. 1–12.
[43] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. *Deep Learning via Semi-supervised Embedding*. Springer, 639–655.
[44] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*. 2048–2057.
[45] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*. 3482–3489.
[46] Carl Yang, Mengxiong Liu, Vincent W. Zheng, and Jiawei Han. 2018. Node, Motif and Subgraph: Leveraging Network Functional Blocks Through Structural Convolution. In *ASONAM*. 1–8.
[47] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*. 40–48.
[48] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. 2016. Minimal gated unit for recurrent neural networks. *IJAC* 13, 3 (2016), 226–234.
[49] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*. 912–919.