Class-Attentive Diffusion Network for Semi-Supervised Classification

Jongin Lim[1], **Daeho Um**[1], **Hyung Jin Chang**[2], **Dae Ung Jo**[1], **Jin Young Choi**[1]
[1] Department of ECE, ASRI, Seoul National University
[2] School of Computer Science, University of Birmingham
[1] {ljin0429,daehoum1,mardaewoon,jychoi}@snu.ac.kr, [2] h.j.chang@bham.ac.uk

Abstract

Recently, graph neural networks for semi-supervised classification have been widely studied. However, existing methods only use the information of limited neighbors and do not deal with the inter-class connections in graphs. In this paper, we propose Adaptive aggregation with Class-Attentive Diffusion (AdaCAD), a new aggregation scheme that adaptively aggregates nodes probably of the same class among K-hop neighbors. To this end, we first propose a novel stochastic process, called Class-Attentive Diffusion (CAD), that strengthens attention to intra-class nodes and attenuates attention to inter-class nodes. In contrast to the existing diffusion methods with a transition matrix determined solely by the graph structure, CAD considers both the node features and the graph structure with the design of our class-attentive transition matrix that utilizes a classifier. Then, we further propose an adaptive update scheme that leverages different reflection ratios of the diffusion result for each node depending on the local class-context. As the main advantage, AdaCAD alleviates the problem of undesired mixing of inter-class features caused by discrepancies between node labels and the graph topology. Built on AdaCAD, we construct a simple model called *Class-Attentive Diffusion Network* (CAD-Net). Extensive experiments on seven benchmark datasets consistently demonstrate the efficacy of the proposed method and our CAD-Net significantly outperforms the state-of-the-art methods. Code is available at *https://github.com/ljin0429/CAD-Net*.

Introduction

Semi-supervised learning is a long-standing problem in machine learning. Many semi-supervised learning algorithms rely on the geometry of the data induced by both labeled and unlabeled data points (Chapelle, Scholkopf, and Zien 2006). Since this geometry can be naturally represented by a graph whose nodes are data points and edges represent relations between data points, graph-based semi-supervised learning has been extensively studied for decades (Zhu, Ghahramani, and Lafferty 2003; Zhou et al. 2004; Belkin, Niyogi, and Sindhwani 2006; Yang, Cohen, and Salakhutdinov 2016; Kipf and Welling 2016). In this paper, we focus on the problem of semi-supervised node classification on graphs.

Graph Neural Networks (GNNs) have achieved remarkable progress in this field recently (Zhou et al. 2018; Wu et al. 2020; Zhang, Cui, and Zhu 2020). In particular, graph convolutions (Kipf and Welling 2016; Gilmer et al. 2017) have received great attention due to its flexibility and good performances. Underlying these methods is a neighborhood aggregation that forms a new representation of a node by aggregating features of itself and its neighbors. The neighborhood aggregation is essentially a type of Laplacian smoothing (Li, Han, and Wu 2018), i.e., making the features of neighboring nodes similar, which makes the subsequent classification task easier. Built on this, several methods have developed the weighted aggregation using attention mechanisms (Veličković et al. 2017; Thekumparampil et al. 2018; Zhang et al. 2018) where the attention weights are determined by the features of each neighboring node pair.

However, one of the fundamental weaknesses with neighborhood aggregation methods is the lack of ability to capture long-range dependencies caused by over-smoothing (Chen et al. 2019). When stacking multiple layers to expand their range, the diameter of the smoothed area grows large, and eventually, the node representations in the entire graph become indistinguishable. Although there have been miscellaneous efforts to overcome this issue (Xu et al. 2018; Abu-El-Haija et al. 2019; Wu et al. 2019), the range of these methods is still limited. Therefore, it is desirable to have the ability to propagate the label information over long-range, especially for large graphs or under sparsely labeled settings.

Recently, diffusion-based methods (Klicpera, Bojchevski, and Günnemann 2018; Jiang et al. 2019; Klicpera, Weißenberger, and Günnemann 2019) have demonstrated the capability of capturing long-range dependencies without leading to over-smoothing. These methods utilize graph diffusion as an alternative to neighborhood aggregation. Graph diffusion is a Markov process which spreads the information from the node to the adjacent nodes at each time step (Masuda, Porter, and Lambiotte 2017). Theoretically, K-steps of feature diffusion means that features of up to K-hop neighbors are aggregated to each node. This aggregation-by-diffusion scheme allows the model to achieve a larger range without changing the neural network, whereas in the neighborhood aggregation scheme expanding the range would require additional layers. However, a major limitation of these methods is that they only utilize the graph structure with a transition matrix of diffusion determined by the graph adjacency matrix. Since edges in real graphs are often noisy (Khan, Ye,

and Chen 2018) and could contain additional information, there exist discrepancies between node labels and the graph structure (Chen et al. 2019), i.e., some nodes may have more inter-class neighbors. Thus, the aggregation scheme determined solely by the graph structure may lead to corrupted representations due to the inter-class connections.

To address the aforementioned limitation, we propose Class-Attentive Diffusion (CAD), a novel stochastic process that strengthens attention to intra-class nodes and attenuates attention to inter-class nodes by considering both the node features and the graph structure. The proposed CAD attentively aggregates nodes probably of the same class among K-hop neighbors so that the feature representations of the same class become similar. Then, we further propose a novel adaptive update scheme that assigns proper reflection ratios of the CAD result for each node depending on the local class-context. If a node has many inter-class neighbors, our adaptation scheme puts more weights on the node's original feature than the aggregated feature by CAD and vice versa. In this work, we call the overall scheme as ***Ada**ptive aggregation with **CAD*** (AdaCAD). Built on AdaCAD, we construct a simple model called *Class-Attentive Diffusion Network* (CAD-Net). Through extensive experiments, we validate the proposed method and show that AdaCAD enables the model to embed more favorable feature representations for better class separation. Our CAD-Net significantly outperforms the state-of-the-art methods on 7 benchmark datasets from 3 different graph domains.

Related Work

Graph Neural Networks

In recent literature on GNNs, there are two mainstreams: spectral-based methods and spatial-based methods. The spectral-based methods (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016) developed graph convolutions in the spectral domain using the graph Fourier transform. However, these methods do not scale well with large graphs due to the computational burden. The spatial-based methods (Niepert, Ahmed, and Kutzkov 2016; Gilmer et al. 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2017; Monti et al. 2017), on the other hand, defined convolution-like operations directly on the graph based on the neighborhood aggregation. The spatial-based methods, in particular, GCN (Kipf and Welling 2016), MPNN (Gilmer et al. 2017), and SAGE (Hamilton, Ying, and Leskovec 2017) have received considerable attention due to its efficiency and superior performance. Built on the neighborhood aggregation scheme, numerous variants have been proposed. In the following, we categorize recent methods into three groups based on what they leverage to improve the model.

(i) *Extended Aggregation*. Mixhop (Abu-El-Haija et al. 2019) concatenates aggregated features from neighbors at different hops before each layer, while in JK (Xu et al. 2018), skip connections are exploited to *jump* knowledge to the last layer. In SGC (Wu et al. 2019), multi-layers of GCN (Kipf and Welling 2016) are simplified into a single layer using the K-th power of an adjacency matrix, which means that the aggregation extends to the K-hop neighbor. These methods use an extended neighborhood for aggregation. However, the range of these methods is still limited, attributed to the low number of layers used.

(ii) *Feature Attention*. Attention-based method such as GAT (Veličković et al. 2017), AGNN (Thekumparampil et al. 2018), and GaAN (Zhang et al. 2018) have utilized attention mechanisms to develop weighted aggregation where the weighting coefficients are determined by the features of each neighboring node pair. However, the aggregation of these methods is still limited to 1-hop neighbor. Meanwhile, Graph U-Nets (Gao and Ji 2019) proposed graph pooling and unpooling operations based on feature attention and then, developed an encoder-decoder architecture in analogy to U-Net (Ronneberger, Fischer, and Brox 2015). However, the pooling operation proposed in their method does not take the graph structure into account but only depends on the node features (Lee, Lee, and Kang 2019).

(iii) *Graph Diffusion*. Recently, there have been several attempts utilizing graph diffusion. These methods aggregate features by propagation over nodes using random walks (Atwood and Towsley 2016; Ying et al. 2018; Ma, Li, and Wang 2019), Personalized PageRank (PPR) (Klicpera, Bojchevski, and Günnemann 2018), Heat Kernel (HK) (Xu et al. 2019), and regularized Laplacian smoothing-based diffusion methods (Jiang et al. 2019). Meanwhile, GDC (Klicpera, Weißenberger, and Günnemann 2019) utilizes generalized graph diffusion (e.g. PPR and HK) to generate a new graph, then use this new graph instead of the original graph to improve performance. However, all of the aforementioned methods do not take node features into account in their diffusion.

Random Walks on Graph

Random walks have been extensively studied in classical graph learning; see (Lovász et al. 1993; Masuda, Porter, and Lambiotte 2017) for an overview of existing methods. In particular, random walks were used in the field of unsupervised node embedding (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Tsitsulin et al. 2018; Abu-El-Haija et al. 2018). Unlike these methods, the proposed method aims to embed a more favorable node representation for semi-supervised classification. To achieve this, the proposed diffusion is class-attentive by considering the node features as well as the graph structure, while in those methods, it only depends on the graph adjacency matrix. In (Wu et al. 2012; Wu, Li, and Chang 2013), Partially Absorbing Random Walk (PARW), a second-order Markov chain with partial absorption at each state, was proposed for semi-supervised learning. Co- & Self-training (Li, Han, and Wu 2018) utilized PARW for label propagation, presenting learning techniques that add highly confident predictions to the training set. However, the state distribution of PARWs is also determined solely by the graph structure. Recently, several methods (Lee, Rossi, and Kong 2018; Akujuobi et al. 2019, 2020) adopted reinforcement learning that aims to learn a policy that attentively selects the next node in the RW process. However, unlike the proposed method, their attention does not explicitly utilize class similarity since they employed additional modules to learn the policy.

Proposed Method

Problem Setup

Formally, the problem of semi-supervised node classification considers a graph $G = (\mathcal{V}, \mathcal{E}, X)$ where $\mathcal{V} = \{v_i\}_{i=1}^N$ is the set of N nodes, $\mathcal{E}$ denotes the edges between nodes, and $X \in \mathbb{R}^{N \times D}$ is a given feature matrix, i.e., x_i, i-th row of X, is D-dimensional feature vector of the node v_i. Since edge attributes may not be given, we consider unweighted version of the graph represented by an adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{N \times N}$ where $a_{ij} = 1$ if $\mathcal{E}(i, j) \neq 0$ and $a_{ij} = 0$ otherwise. We denote the given label set as Y_L associated with the labeled node set $\mathcal{V}_L$, i.e., $y_i \in Y_L$ be an one-hot vector indicating one of C classes for v_i. We focus on the transductive setting (Yang, Cohen, and Salakhutdinov 2016) which aims to infer the labels of the remaining unlabeled nodes based on (X, A, Y_L).

In general, the model for semi-supervised node classification can be expressed as

$$Z = f_\theta(X, A) \quad \text{and} \quad \hat{y}_i = g_\phi(z_i) \;\; (i = 1, 2, \cdots, N) \tag{1}$$

where f_θ is a feature embedding network to embed the feature representations Z from (X, A), and g_ϕ is a node classifier predicting $\hat{y}_i$ from z_i, i-th row of Z. The feature embedding network f_θ is realized by GNNs in recent literature. The process of GNN can be decomposed into two steps: feature transformation and feature aggregation where the former stands for a non-linear transformation of node features and the latter refers to the process of forming new representations via aggregating proximal node features.

In this paper, we focus on the process of feature aggregation. More specifically, we aim to design an aggregation scheme which can be applied right before the classifier from Eq. (1) to embed more favorable feature representations for class separation. To this end, we first propose a novel Class-Attentive Diffusion (CAD), which attentively aggregates nodes probably of the same class among K-hop neighbors so that the representations of the same class become similar. Given $Z \in \mathbb{R}^{N \times F}$, CAD produces new feature representations $Z^{(\mathrm{CAD})} \in \mathbb{R}^{N \times F}$ as follows,

$$Z^{(\mathrm{CAD})} \leftarrow \mathrm{CAD}(Z, A, \{g_\phi(z_i)\}_{i=1}^N). \tag{2}$$

Note that the node features (Z), the graph structure (A), and the class information (g_ϕ) are jointly utilized. Then, we further propose **Ada**ptive aggregation with **CAD** (AdaCAD) that leverages different reflection ratios of the diffusion result for each node depending on the local class-context. That is, AdaCAD produces the final feature representations $Z^{(\mathrm{AdaCAD})} \in \mathbb{R}^{N \times F}$ as follows,

$$Z^{(\mathrm{AdaCAD})} \leftarrow \mathrm{AdaCAD}(Z, Z^{(\mathrm{CAD})}, \Gamma) \tag{3}$$

where Γ assigns proper weights between Z and $Z^{(\mathrm{CAD})}$ for each node. Lastly, built on AdaCAD, we present a simple model called *Class-Attentive Diffusion Network*.

Class-Attentive Diffusion

In this section we present a novel stochastic process called Class-Attentive Diffusion (CAD), which combines the advantages of both the attention mechanism and the diffusion process. The proposed CAD consists of N Class-Attentive Random Walks (CARWs) starting at each node in the graph. For clarity, we first explain how a single CARW is defined.

Suppose a CARW that starts from the node v_i. The walker determines the next node among the neighbor by comparing the class likelihood given the node features, i.e., comparing $\mathbf{p}_i = p(y_i|z_i)$ and $\mathbf{p}_j = p(y_j|z_j)$ for $j \in \mathcal{N}(i)$. Our design objective is that the more similar $\mathbf{p}_i$ and $\mathbf{p}_j$, the more likely the walker moves from v_i to v_j. To this end, we define the transition probability from v_i to v_j as

$$T_{ij} = \mathrm{softmax}_{j \in \mathcal{N}(i)}(\mathbf{p}_i^T \mathbf{p}_j). \tag{4}$$

Note that, $\mathbf{p}_i$ is a categorical distribution of which c-th element $\mathbf{p}_i(c)$ is the probability of node i belongs to class c. Thus, the cosine distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ (i.e., $\mathbf{p}_i^T \mathbf{p}_j$) can be one possible solution for measuring the similarity between them. However, the true class likelihood $\mathbf{p}_i$ is intractable. Instead, we approximate the true distribution by exploiting the classifier g_ϕ in Eq. (1) where the probability of each class is inferred by g_ϕ based on the node feature z_i. That is,

$$\mathbf{p}_i \approx p(\hat{y}_i|z_i) = g_\phi(z_i). \tag{5}$$

As the learning progresses, the transition matrix in Eq. (4) gradually becomes more class-attentive by means of g_ϕ. This is the key difference from the recent diffusion-based methods, APPNP (Klicpera, Bojchevski, and Günnemann 2018), GDEN (Jiang et al. 2019), and GDC (Klicpera, Weißenberger, and Günnemann 2019), where the transition matrix is determined solely by the adjacency matrix.

Let a row vector $\pi_i^{(t)} \in \mathbb{R}^N$ be the state distribution of the CARW after t steps. This can be naturally derived by a Markov chain, i.e., $\pi_i^{(t+1)} = \pi_i^{(t)} T$, where the initial state distribution $\pi_i^{(0)}$ be a one hot vector indicating the starting node v_i. Then, this can be naturally extended to CAD where $\mathbf{\Pi}^{(K)} \in \mathbb{R}^{N \times N}$ be the state distribution matrix after K-steps of CAD with entries $\mathbf{\Pi}^{(K)}(i, j) = \pi_i^{(K)}(j)$.

Now, we can define a new aggregation method with K-steps of CAD, which forms a new feature representation of the node v_i as follows,

$$z_i^{(\mathrm{CAD})} = \sum_j \pi_i^{(K)}(j) \cdot z_j. \tag{6}$$

Note that $\pi_i^{(K)}(j)$ is zero for v_j beyond K-hop from v_i. Hence, $\pi_i^{(K)}(j)$ naturally reflects the class similarity as it grows with the similarity between $\mathbf{p}_i$ and $\mathbf{p}_j$. That is, $z_i^{(\mathrm{CAD})}$ is essentially an attentive aggregation of K-hop neighbors where CAD strengthens attention to intra-class nodes and attenuates attention to inter-class nodes.

Adaptive Aggregation with CAD

In this section, we present Adaptive aggregation with CAD (AdaCAD). We start by introducing our motivation. In real graphs, some nodes may be connected to nodes of various classes, or even worse, nodes of the same class may not even exist in their neighbors. Intuitively, in these cases, aggregated features from neighbors may lead to corrupted representations due to the inter-class connections. Therefore, it

should be needed to adaptively adjust the degree of aggregation for each node depending on its local class-context.

Motivated by this, we define AdaCAD to form a new feature representation of the node v_i as follows,

$$z_i^{(\text{AdaCAD})} = (1-\gamma_i)\cdot z_i + \gamma_i \cdot z_i^{(\text{CAD})}. \tag{7}$$

Here, $\gamma_i \in [0,1]$ controls the trade-off between its own node feature z_i and the aggregated feature $z_i^{(\text{CAD})}$ from Eq. (6) by considering the local class-context of v_i. For the node with neighbors of the same class, γ_i should be a large value to accelerate proper smoothing. In the opposite situation, γ_i should be adjusted to a small value to preserve its original feature and avoid undesired smoothing.

To this end, we define a control variable c_i as

$$c_i = \frac{1}{\deg(i)} \sum_{j \in \mathcal{N}(i)} g_\phi(z_i)^T g_\phi(z_j) \tag{8}$$

where $\deg(i)$ is the degree of v_i and g_ϕ is the aforementioned classifier. Then, the range of c_i would be $0 \le c_i \le 1$. The meaning of c_i is that the more nodes of the same class in the neighborhood, the greater the value of c_i and vice versa. Therefore, we set up an adaptive formula for γ_i as

$$\gamma_i = (1-\beta)c_i + \beta\gamma_u \tag{9}$$

where $\gamma_u = 1$ is the upper bound of γ_i to keep $0 \le \gamma_i \le 1$ for interpolation of each node feature and the diffusion result. Note that γ_i divides c_i and γ_u internally in the ratio of $\beta : (1-\beta)$ where $\beta \in [0,1]$ controls the sensitivity of how much γ_i will be adjusted according to c_i. Since different graphs exhibit different neighborhood structures (Klicpera, Bojchevski, and Günnemann 2018), the sensitivity β is determined empirically for each dataset.

Now, we conclude the section with the overall formula of the proposed AdaCAD in a matrix form. By letting $\mathbf{\Gamma} = \text{diag}(\gamma_1, \gamma_2, \cdots, \gamma_N)$ and combining Eq. (6) and (7) together, the entire aggregation scheme of AdaCAD can be expressed as follows,

$$Z^{(\text{AdaCAD})} = (\mathbf{I} - \mathbf{\Gamma}) \cdot Z + \mathbf{\Gamma} \cdot \mathbf{\Pi}^{(K)} \cdot Z \tag{10}$$

where $\mathbf{\Pi}^{(K)}$ is the state distribution matrix after K-steps of CAD. Note that AdaCAD does not require additional learning parameters since we utilize the classifier g_ϕ.

Class-Attentive Diffusion Network

Built on AdaCAD, we construct *Class-Attentive Diffusion Network* (CAD-Net) for semi-supervised classification. CAD-Net consists of the feature embedding network f_θ followed by AdaCAD and the classifier g_ϕ as defined in Eq. (1), (3) and (10). More specifically, we realize f_θ with 2-layers of MLP for simplicity, as the process of feature aggregation can be sufficiently performed in AdaCAD, and g_ϕ is realized by the softmax function, i.e., $\hat{y_i} = g_\phi(z_i) = \text{softmax}(z_i)$ as in other literature (Kipf and Welling 2016; Wu et al. 2019; Jiang et al. 2019), and thus the dimension of z_i is set to the number of classes. The whole network parameters can then be trained in an end-to-end manner by minimizing the cross-entropy loss function $\mathcal{L}_{\text{sup}}$ over all labeled nodes. By minimizing the cross-entropy between the label y_i and the prediction $\hat{y_i}$ for all $y_i \in Y_L$, the model can be learned to enhance the element of z_i that corresponds to the index indicating the class of y_i, which facilitates the class separation. In addition to $\mathcal{L}_{\text{sup}}$, we consider another regularization objective. As defined in Eq. (4) and (5), the transition matrix of CAD is determined by $\mathbf{p}_i$ where the initial distribution $\mathbf{p}_i$ for each node should generally be close to a one-hot vector such that the resulting transition matrix becomes more class-attentive. Thus, we regularize the entropy of $\mathbf{p}_i$ by minimizing $\mathcal{L}_{\text{ent}} = \sum_{i=1}^{N} H(\mathbf{p}_i)$ where H denotes the entropy function. During training, $\mathcal{L}_{\text{sup}}$ and $\mathcal{L}_{\text{ent}}$ are jointly minimized by using Adam optimizer (Kingma and Ba 2014). We report the detailed implementation in Appendix A.[1]

Dataset	Nodes	Edges	Features	Classes
CITESEER	3327	4552	3703	6
CORA	2708	5278	1433	7
PUBMED	19717	44324	500	3
AMAZON COMP.	13752	245861	767	10
AMAZON PHOTO	7650	119081	745	8
COAUTHOR CS	18333	81894	6805	15
COAUTHOR PHY.	34493	247962	8415	5

Table 1: Dataset statistics.

Experiments

Datasets

We conducted experiments on 7 benchmark datasets from 3 different graph domains: *Citation Networks* (CiteSeer, Cora, and PubMed), *Recommendation Networks* (Amazon Computers and Amazon Photo), and *Co-authorship Networks* (Coauthor CS and Coauthor Physics). **CiteSeer**, **Cora**, and **PubMed** are citation networks where each node represents a document and each edge represents a citation link. Node features are bag-of-words descriptors of the documents, and class labels are given by the document's fields of study. **Amazon Computers** and **Amazon Photo** are segments of Amazon co-purchase graph. Here, each node represents a product and each edge indicates that two goods are frequently bought together. Node features are bag-of-words descriptors which encode the product reviews, and class labels are given by the product category. **Coauthor CS** and **Coauthor Physics** are co-authorship networks based on MS Academic Graph where each node represents an author and an edge is connected if they have co-authored a paper. Node features represent paper keywords for each author's papers, and class labels indicate the most active fields of study for each author. Table 1 summarizes the dataset statistics.

Experimental Setup

For citation networks, we followed the standard benchmark setting suggested in (Yang, Cohen, and Salakhutdinov 2016). We evaluated on the same train/validation/test split, which uses 20 nodes per class for train, 500 nodes for validation, and 1000 nodes for test. For the credibility of the

[1] *https://github.com/ljin0429/CAD-Net*

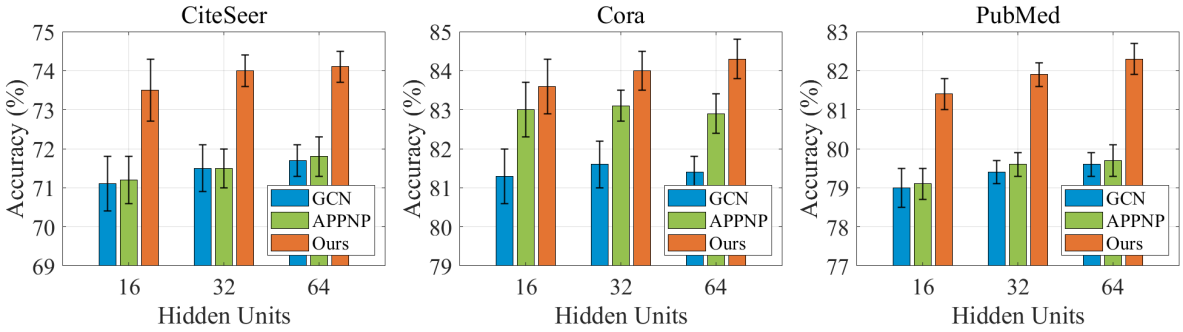

Figure 1: Accuracy (%) with different hidden units. Note that GCN, APPNP, and the proposed CAD-Net have the same number of parameters. For all datasets, CAD-Net significantly outperform GCN and APPNP with the same number of parameters.

results, we report the average accuracy (%) with the standard deviation evaluated on 100 independent runs.

For recommendation and co-authorship networks, we closely followed the experimental setup in (Chen et al. 2019). We used 20 nodes per class for train, 30 nodes per class for validation, and the rest nodes for test. We randomly split the nodes and report the average accuracy (%) with the standard deviation evaluated on 100 random splits.

We compared the proposed method with the following 12 state-of-the-art methods: **Cheby** (Defferrard, Bresson, and Vandergheynst 2016), **GCN** (Kipf and Welling 2016), **SAGE** (Hamilton, Ying, and Leskovec 2017), **JK** (Xu et al. 2018), **MixHop** (Abu-El-Haija et al. 2019), **SGC** (Wu et al. 2019), **AGNN** (Thekumparampil et al. 2018), **GAT** (Veličković et al. 2017), **Graph U-Nets** (Gao and Ji 2019), **APPNP** (Klicpera, Bojchevski, and Günnemann 2018), **GDC** (Klicpera, Weißenberger, and Günnemann 2019), and **GDEN** (Jiang et al. 2019). In all experiments, the publicly released codes were employed.

Model Analysis

In this section, we provide comprehensive analysis of the proposed method on CiteSeer, Cora, and PubMed as they are the most widely used benchmark datasets in the literature.

Influence of AdaCAD. To verify the effectiveness of AdaCAD, we compared AggCAD with 7 different aggregation methods. For a fair comparison, only AdaCAD is replaced with the same CAD-Net architecture. Firstly, we consider 4 diffusion methods including Random Walks (RW), symmetric Normalized Adjacency matrix (symNA), Personalized PageRank (PPR) (Page et al. 1999), and Heat Kernel (HK) (Kondor and Lafferty 2002). For RW and symNA, the transition matrix is defined as $D^{-1}A$ and $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ respectively, and we proceed K-steps of feature diffusion according to their transition matrix. For PPR and HK, the closed-form solution of the diffusion state distribution is used as in (Klicpera, Weißenberger, and Günnemann 2019). Secondly, we consider 2 attentive diffusion variants. To the best of our knowledge, CAD is the first attempt that incorporates the feature attention and the diffusion process. Therefore, we construct GAT+RW and TF+RW based on GAT (Veličković et al. 2017) and Transformer (Vaswani et al. 2017) respectively. In GAT+RW, the transition is defined by the attention value computed by GAT, and we proceed K-steps of feature diffusion according to it. Likewise, in TF+RW, the transition is defined by Transformer-style attention, i.e., $T_{ij} = \mathrm{softmax}_j(f_{\mathrm{Q}}(z_i)^T f_{\mathrm{K}}(z_j))$. Lastly, we consider the model that only uses CAD for aggregation.

	CITESEER	CORA	PUBMED
RW	71.5 ± 0.5	82.4 ± 0.5	79.6 ± 0.4
symNA	71.4 ± 0.6	82.1 ± 0.5	79.8 ± 0.3
PPR	72.5 ± 0.9	82.8 ± 0.6	79.3 ± 0.6
HK	71.8 ± 0.5	82.3 ± 0.6	79.4 ± 0.5
GAT+RW	70.2 ± 1.4	81.3 ± 1.4	77.7 ± 1.0
TF+RW	70.4 ± 1.5	82.8 ± 1.2	78.6 ± 1.1
CAD	73.5 ± 0.5	83.7 ± 0.5	80.2 ± 0.5
AdaCAD	$\mathbf{74.1 \pm 0.4}$	$\mathbf{84.3 \pm 0.5}$	$\mathbf{82.3 \pm 0.4}$

Table 2: Accuracy (%) with different aggregation methods. Note that only the aggregation part (AdaCAD) is switched from the same CAD-Net architecture.

Table 2 shows the overall comparisons with the aforementioned variants. Compared to RW, sym, PPR, and HK, which only utilize the graph structure, our variants (CAD-only and AggCAD) show superior results. The better performance comes from the proposed class-attentive transition matrix both utilizing node features and the graph structure. While GAT+RW and TF+RW can utilize both node features and the graph structure, the performances are not sufficient, which demonstrate the effectiveness of our design of class-attentive diffusion. Lastly, AdaCAD shows better performance than only using CAD. By means of Γ in AdaCAD, the model prevents undesired mixing from inter-class neighbors, which provides additional performance gains to CAD.

Influence of Hidden Units. Unlike attention-based methods (AGNN and GAT), the proposed CAD can be self-guided by the classifier without the need for additional parameters for attention. Thus, the total number of parameters can be implemented in the same way as the vanilla GCN. To validate the effectiveness of AdaCAD, we evaluated the performance across the different numbers of hidden units in the feature embedding network f_θ, and compared the re-

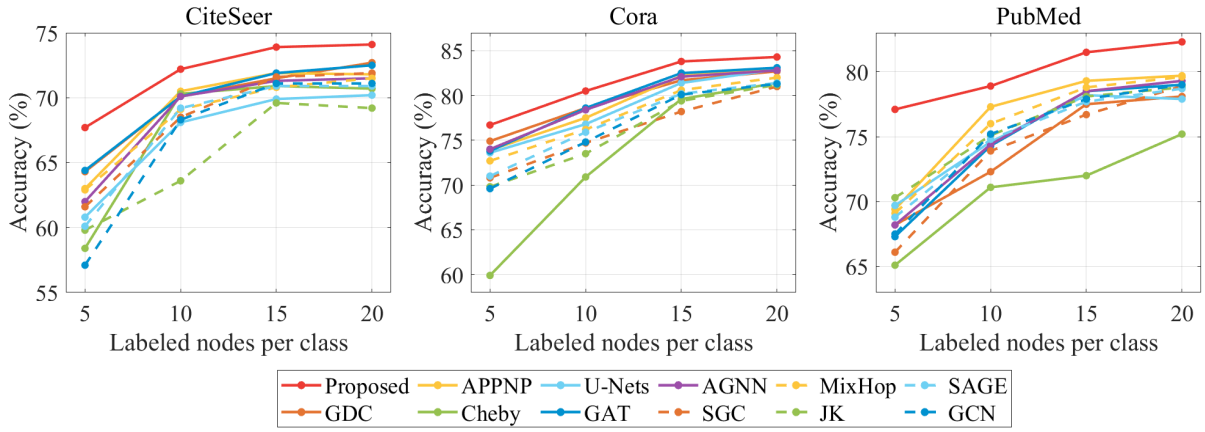

Figure 2: Accuracy (%) with different numbers of labeled nodes per class. The proposed CAD-Net shows robust and superior performance for all settings.

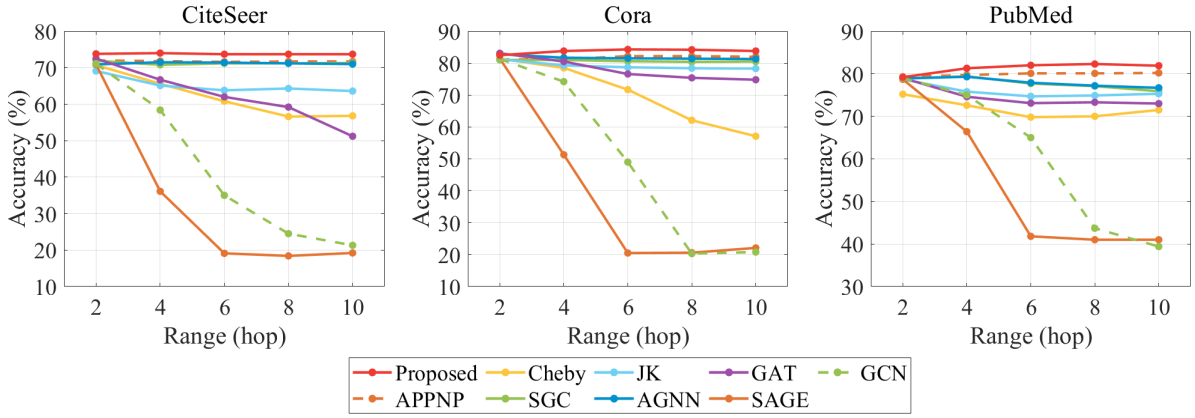

Figure 3: Accuracy (%) with varying ranges of the model. The proposed CAD-Net shows superior performance regardless of the different ranges. The dominance of CAD-Net increases for a longer range.

sults with GCN and APPNP which have the same number of parameters. As shown in Figure 1, CAD-Net shows robust performance with respect to the number of hidden units. Further, for all experiments, we can observe that CAD-Net significantly outperforms GCN and APPNP with the same number of parameters. This demonstrates that the superior performance of CAD-Net is attributed to the proposed AdaCAD, not the power of the feature embedding network.

Influence of β. We also analyzed the influence of the hyperparameter β which controls the sensitivity of how much γ will be adjusted. Due to the space limit, we attach the results to Appendix B.4. While the optimum differs slightly for each dataset, we consistently found that any $\beta \in [0.65, 0.95]$ achieves the state-of-the-art performances.

Different Label Rates. We then explored how the number of training nodes per class impacts the accuracy of the models. The ability to maintain robust performance even under very sparsely labeled settings is important. We compared the performances when the number of labeled nodes per class is changed to 20, 15, 10, and 5. The overall results are presented in Figure 2. CAD-Net shows robust and superior performance even under the very sparsely labeled setting and outperforms all other methods. Note that, the diffusion-based methods (APPNP and GDC) do not show satisfactory results despite their wide range. This is because these methods only utilize the graph structure. In contrast, the proposed method aggregates nodes from a wide range and the importance of each node reflects both node features and the graph structure, which contributes to the superior performances of CAD-Net. Especially, the superiority of CAD-Net is more obvious in PubMed which is a large dataset. This further demonstrates the effectiveness of the proposed method.

Different Ranges. Figure 3 shows influence of the different ranges. As expected, the neighborhood aggregation methods degrade performance with increasing layers. While the diffusion-based methods maintain the performance with increasing ranges, CAD-Net shows superior performance for all ranges. Also, as in the previous experiment, the superiority of CAD-Net is particularly evident in PubMed, which suggests that the proposed method is able to accommodate larger graphs or sparsely labeled settings.

Type	Method	CITESEER	CORA	PUBMED	AMAZON COMP.	AMAZON PHOTO	COAUTHOR CS	COAUTHOR PHYSICS
Vanilla	Cheby	70.7 ± 0.5	81.4 ± 0.5	75.2 ± 1.4	76.2 ± 2.4	85.9 ± 2.3	OOM	OOM
	GCN	71.1 ± 0.7	81.3 ± 0.7	79.0 ± 0.5	78.7 ± 3.3	88.9 ± 1.9	91.3 ± 0.6	93.3 ± 0.8
	SAGE	70.9 ± 0.7	81.4 ± 0.7	78.7 ± 0.4	78.9 ± 2.1	89.4 ± 1.8	91.6 ± 0.6	93.1 ± 0.8
Extended Aggregation	JK	69.1 ± 1.1	81.2 ± 0.8	78.7 ± 0.5	79.0 ± 3.7	89.1 ± 2.0	91.7 ± 0.5	93.2 ± 0.9
	MixHop	71.5 ± 0.8	82.0 ± 1.0	79.4 ± 0.5	79.5 ± 2.8	88.8 ± 1.7	OOM	OOM
	SGC	71.9 ± 0.1	81.0 ± 0.2	78.9 ± 0.1	81.5 ± 1.8	90.0 ± 1.5	91.2 ± 0.6	92.9 ± 1.0
Feature Attention	AGNN	71.5 ± 0.7	82.8 ± 0.6	79.3 ± 0.8	73.5 ± 2.7	88.0 ± 3.4	92.1 ± 0.6	93.8 ± 0.7
	GAT	72.5 ± 0.8	83.1 ± 0.8	79.0 ± 0.3	80.5 ± 2.2	90.6 ± 1.2	91.0 ± 0.5	93.1 ± 0.6
	U-Nets	70.2 ± 1.0	82.9 ± 0.7	78.0 ± 0.5	76.9 ± 2.1	87.1 ± 1.8	91.7 ± 1.0	93.4 ± 0.7
Graph Diffusion	APPNP	71.8 ± 0.5	82.9 ± 0.5	79.7 ± 0.3	81.0 ± 1.9	90.5 ± 1.6	92.3 ± 0.4	93.5 ± 0.7
	GDC	72.7 ± 0.8	82.7 ± 0.7	78.1 ± 0.3	81.6 ± 2.8	88.8 ± 1.7	OOM	92.7 ± 0.8
	*GDEN	72.8	82.0	78.7	-	-	-	-
Proposed	CAD-Net	$\mathbf{74.1 \pm 0.4}$	$\mathbf{84.3 \pm 0.5}$	$\mathbf{82.3 \pm 0.4}$	$\mathbf{82.1 \pm 2.0}$	$\mathbf{90.9 \pm 1.5}$	$\mathbf{93.5 \pm 0.6}$	$\mathbf{94.7 \pm 0.4}$

Table 3: Accuracy (%) under standard benchmark setting. For all experiments, we report the performance evaluated over 100 independent runs. OOM denotes out-of-memory. (*We report the numbers taken from their paper since the code is not available.)

Method	CITESEER	CORA	PUBMED
Cheby	69.1 ± 1.9	77.8 ± 2.3	73.1 ± 3.2
GCN	68.3 ± 1.9	79.3 ± 1.8	77.2 ± 2.6
SAGE	68.8 ± 1.8	79.8 ± 1.7	77.2 ± 2.5
JK	67.5 ± 1.9	78.6 ± 1.9	77.7 ± 2.7
MixHop	68.4 ± 1.7	80.7 ± 1.7	77.8 ± 2.5
SGC	69.2 ± 1.7	79.9 ± 1.8	77.0 ± 2.6
AGNN	69.4 ± 1.8	80.8 ± 1.8	78.0 ± 2.4
GAT	70.0 ± 1.9	81.6 ± 1.5	77.3 ± 2.4
U-Nets	67.9 ± 1.9	81.2 ± 1.9	77.8 ± 2.6
APPNP	69.9 ± 1.7	81.8 ± 1.5	78.8 ± 2.5
GDC	70.9 ± 1.7	81.9 ± 1.5	76.9 ± 2.4
CAD-Net	$\mathbf{71.1 \pm 1.6}$	$\mathbf{82.4 \pm 1.4}$	$\mathbf{79.6 \pm 2.4}$

Table 4: Average accuracy (%) evaluated on 100 *Random* train/validation/test splits.

Comparison with State-of-the-art Methods

Evaluation on Benchmark Datasets. Table 3 shows the overall results under standard benchmark settings. In all experiments, the proposed CAD-Net shows superior performance to other methods. We also provide statistical analysis of the results in Appendix B.6, demonstrating that CAD-Net achieves statistically significant improvements. The better performance of CAD-Net comes from the proposed adaptive aggregation scheme based on the class-attentive diffusion both utilizing node features and the graph structure in the transition matrix. In addition, we provide further comparisons with the latest methods (Liu, Gao, and Ji 2020; Chen et al. 2020; Hassani and Khasahmadi 2020; Zhu et al. 2020; Zhang et al. 2020) and our CAD-Net still achieves state-of-the-art performance (see Appendix B.7).

Computational Complexity. In terms of memory requirement, CAD-Net is as efficient as APPNP with the same number of parameters (see Figure 1). Only one forward operation is additionally required to obtain our class-attentive transition probability. To further validate the computational efficiency of CAD-Net, we compared the average training time per epoch (ms) measured on a single Nvidia GTX 1080 Ti machine. As expected, we confirmed that CAD-Net is on par with APPNP and much faster than GAT. The detailed results are provided in Appendix B.8.

Random Splits. Recently, (Shchur et al. 2018) pointed out that the data split (train, validation, test) has a significant influence on the performance. Therefore, we further evaluated average accuracy computed over 100 *Random* splits where the splits are *randomly* drawn with 20 nodes per class for train, 500 nodes for validation, and 1000 nodes for test. As shown in Table 4, CAD-Net shows robust and superior performance regardless of the data splits.

Conclusion

In this paper, we propose Adaptive aggregation with Class-Attentive Diffusion (AdaCAD), a new aggregation scheme for semi-supervised classification on graphs. The main benefits of the proposed AdaCAD are three aspects. (i) AdaCAD attentively aggregates nodes probably of the same class among K-hop neighbors employing a novel Class-Attentive Diffusion (CAD). Unlike the existing diffusion methods, both the node features and the graph structure are leveraged in CAD with the design of the class-attentive transition matrix which utilizes the classifier. (ii) For each node, AdaCAD adjusts the reflection ratio of the diffusion result differently depending on the local class-context, which prevents undesired mixing from inter-class neighbors. (iii) AdaCAD is computationally efficient and does not require additional learning parameters since the class-attentive transition probability is defined by the classifier. Extensive experimental results demonstrate the validity of AdaCAD and Class-Attentive Diffusion Network (CAD-Net), our simple model based on AdaCAD, achieves state-of-the-art performances by a large margin on seven benchmark datasets.

Acknowledgments

This research was supported by the IITP (Institute for Information & Communication Technology Promotion) grant funded by the MSIT (Ministry of Science and ICT, Korea): [2017-0-00306, Outdoor Surveillance Robots] and [IITP-2020-2020-0-01789, ITRC(Information Technology Research Center) support program].

Ethics Statement

Graphs accommodate many potential real-world applications such as social networks and web pages. Our research is a study of neural networks applicable in the graph domain. Therefore, our research can be an important basis for graph-based applications to be applied in real life in the future. Besides, a large amount of cost is required to acquire high quality of labeled data. The problem of semi-supervised learning, which we focus on, can secure robust performance with a small number of labeled data, thus contributing to lowering the threshold of solving industrial or social problems using machine learning at a low cost.

References

Abu-El-Haija, S.; Perozzi, B.; Al-Rfou, R.; and Alemi, A. A. 2018. Watch your step: Learning node embeddings via graph attention. In *Advances in Neural Information Processing Systems*, 9180–9190.

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Harutyunyan, H.; Alipourfard, N.; Lerman, K.; Steeg, G. V.; and Galstyan, A. 2019. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. *arXiv preprint arXiv:1905.00067* .

Akujuobi, U.; Yufei, H.; Zhang, Q.; and Zhang, X. 2019. Collaborative graph walk for semi-supervised multi-label node classification. In *2019 IEEE International Conference on Data Mining (ICDM)*, 1–10. IEEE.

Akujuobi, U.; Zhang, Q.; Yufei, H.; and Zhang, X. 2020. Recurrent Attention Walk for Semi-supervised Classification. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 16–24.

Atwood, J.; and Towsley, D. 2016. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, 1993–2001.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7(Nov): 2399–2434.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* .

Chapelle, O.; Scholkopf, B.; and Zien, A. 2006. *Semi-supervised learning*. Cambridge, USA: MIT Press.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2019. Measuring and Relieving the Over-smoothing Problem for Graph Neural Networks from the Topological View. *arXiv preprint arXiv:1909.03211* .

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 1725–1735.

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

Gao, H.; and Ji, S. 2019. Graph u-nets. *arXiv preprint arXiv:1905.05178* .

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

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive Multi-View Representation Learning on Graphs. *arXiv preprint arXiv:2006.05582* .

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* .

Jiang, B.; Lin, D.; Tang, J.; and Luo, B. 2019. Data Representation and Learning With Graph Diffusion-Embedding Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10414–10423.

Khan, A.; Ye, Y.; and Chen, L. 2018. On Uncertain Graphs. *Synthesis Lectures on Data Management* 10(1): 1–94.

Kingma, D.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

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

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

Klicpera, J.; Weißenberger, S.; and Günnemann, S. 2019. Diffusion Improves Graph Learning. In *Advances in Neural Information Processing Systems*, 13333–13345.

Kondor, R. I.; and Lafferty, J. 2002. Diffusion kernels on graphs and other discrete structures. In *ICML*.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. *arXiv preprint arXiv:1904.08082* .

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

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 338–348.

Lovász, L.; et al. 1993. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty* 2(1): 1–46.

Ma, Z.; Li, M.; and Wang, Y. 2019. PAN: Path integral based convolution for deep graph neural networks. *arXiv preprint arXiv:1904.10996* .

Masuda, N.; Porter, M. A.; and Lambiotte, R. 2017. Random walks and diffusion on networks. *Physics reports* 716: 1–58.

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5115–5124.

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

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

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

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* .

Thekumparampil, K. K.; Wang, C.; Oh, S.; and Li, L.-J. 2018. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735* .

Tsitsulin, A.; Mottin, D.; Karras, P.; and Müller, E. 2018. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*, 539–548.

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

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

Wu, F.; Zhang, T.; Souza Jr, A. H. d.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153* .

Wu, X.-M.; Li, Z.; and Chang, S.-F. 2013. Analyzing the harmonic structure in graph-based learning. In *Advances in Neural Information Processing Systems*, 3129–3137.

Wu, X.-M.; Li, Z.; So, A. M.; Wright, J.; and Chang, S.-F. 2012. Learning with partially absorbing random walks. In *Advances in neural information processing systems*, 3077–3085.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* .

Xu, B.; Shen, H.; Cao, Q.; Cen, K.; and Cheng, X. 2019. Graph convolutional networks using heat kernel for semi-supervised learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1928–1934. AAAI Press.

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

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* .

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 974–983.

Zhang, J.; Shi, X.; Xie, J.; Ma, H.; King, I.; and Yeung, D.-Y. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294* .

Zhang, K.; Zhu, Y.; Wang, J.; and Zhang, J. 2020. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations*.

Zhang, Z.; Cui, P.; and Zhu, W. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* .

Zhou, D.; Bousquet, O.; Lal, T. N.; Weston, J.; and Schölkopf, B. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*, 321–328.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* .

Zhu, H.; Feng, F.; He, X.; Wang, X.; Li, Y.; Zheng, K.; and Zhang, Y. 2020. Bilinear graph neural network with neighbor interactions. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 5.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 912–919.