Label Efficient Semi-Supervised Learning via Graph Filtering

Qimai Li[1] Xiao-Ming Wu*[1] Han Liu[1] Xiaotong Zhang[1] Zhichao Guan[12]

[1]The Hong Kong Polytechnic University, [2]Zhejiang University,
{csqmli,csxmwu,cshliu,csxtzhang}@comp.polyu.edu.hk, zcguan@zju.edu.cn

Abstract

Graph-based methods have been demonstrated as one of the most effective approaches for semi-supervised learning, as they can exploit the connectivity patterns between labeled and unlabeled data samples to improve learning performance. However, existing graph-based methods either are limited in their ability to jointly model graph structures and data features, such as the classical label propagation methods, or require a considerable amount of labeled data for training and validation due to high model complexity, such as the recent neural-network-based methods. In this paper, we address label efficient semi-supervised learning from a graph filtering perspective. Specifically, we propose a graph filtering framework that injects graph similarity into data features by taking them as signals on the graph and applying a low-pass graph filter to extract useful data representations for classification, where label efficiency can be achieved by conveniently adjusting the strength of the graph filter. Interestingly, this framework unifies two seemingly very different methods – label propagation and graph convolutional networks. Revisiting them under the graph filtering framework leads to new insights that improve their modeling capabilities and reduce model complexity. Experiments on various semi-supervised classification tasks on four citation networks and one knowledge graph and one semi-supervised regression task for zero-shot image recognition validate our findings and proposals.

1. Introduction

The success of deep learning and neural networks comes at the cost of large amounts of labeled data and long training time. Semi-supervised learning [13] is important as it can leverage ample available unlabeled data to aid supervised learning, thus greatly saving the cost, trouble, and time for human labeling. Many researches have shown that when used properly, unlabeled data can significantly improve learning performance [64, 31, 32].

One effective approach to making use of unlabeled data is to represent the data in a graph where each labeled or unlabeled sample is a vertex and then model the relations between vertices. For some applications such as social network analysis, data exhibits a natural graph structure. For some other applications such as image or text classification, data may come in a vector form, and a graph is usually constructed using data features. Nevertheless, in many cases, graphs only partially encode data information. Take document classification in a citation network as an example, the citation links between documents form a graph which represents their citation relation, and each document is represented as a bag-of-words feature vector which describes its content. To correctly classify a document, both the citation relations and the content information need to be taken into account, as they contain different aspects of document information. For graph-based semi-supervised learning, the key challenge is to exploit graph structures as well as other information especially data features to improve learning performance.

Despite many progresses, existing methods are still limited in their capabilities to leverage multiple modalities of data information for learning. The classical label propagation methods only exploit graph structures to make predictions on unlabeled examples, which are often inadequate in many scenarios. To go beyond their limit and jointly model graph structures and data features, a common approach is to train a supervised learner to classify data features while regularizing the classifier with graph similarity. Manifold regularization [4] trains a support vector machine with a graph Laplacian regularizer. Deep semi-supervised embedding [53] and Planetoid [56] train a neural network with an embedding-based regularizer. Recently, graph convolutional networks (GCN) [32] have demonstrated impressive results in semi-supervised learning, due to its special design of a first-order convolutional filter that nicely integrates graph and feature information in each layer. The success of

*Corresponding author

GCN has inspired many follow-up works [60, 62] on graph neural networks for semi-supervised learning. However, although these neural-network-based models tend to have stronger modelling capabilities than the conventional ones, they typically require a considerable amount of labeled data for training and validation due to high model complexity, hence may not be label efficient.

In this paper, we propose to study semi-supervised learning from a principled graph filtering perspective. The basic idea is to take data features as signals sitting on the underlying graph that encodes relations between data samples, and uses the graph to design proper low-pass graph convolutional filters to generate smooth and representative features for subsequent classification. In this process, graph similarity is injected into data features to produce more faithful data representations. It also enables learning with few labels by flexibly adjusting filter strength to achieve label efficiency. More interestingly, it unifies well-known semi-supervised learning methods including the label propagation methods [61] and the graph convolutional networks [32], with useful insights for improving their modelling capabilities.

Under the graph filtering framework, we show that label propagation methods can be decomposed into three components: graph signal, graph filter, and classifier. Based on this, we then propose a class of generalized label propagation (GLP) methods by naturally extending the three components, including using data feature matrix instead of label matrix as input graph signals, extending the graph filter to be any low-pass graph convolutional filter, and using any desired classifier for classification. GLP can achieve label efficiency in semi-supervised learning by taking advantages of data features, strong and efficient graph filters, and powerful supervised classifiers.

The popular graph convolutional networks (GCN) [32] can also be interpreted under the graph filtering framework. It has been shown that GCN implements the graph convolution in each layer by conducting Laplacian smoothing [33]. When revisited under the graph filtering framework, it further elucidates the inner working of GCN including the renormalization trick and the model parameter settings. Furthermore, it leads to an improved GCN model (IGCN) that is more flexible and label efficient. By adding an exponent parameter on the filter of GCN to easily control filter strength, IGCN can significantly reduce trainable parameters and effectively prevent overfitting when training data is very limited.

We conduct extensive experiments to validate our findings and the effectiveness of the proposed methods. We test a variety of semi-supervised classification tasks including document classification on four citation networks and entity classification on one knowledge graph. We also test a semi-supervised regression task for zero-shot image recognition. The proposed GLP and IGCN methods perform superiorly in terms of prediction accuracy and training efficiency.

Our contributions are summarized as follows. First, we propose a graph filtering framework for semi-supervised learning, which provides new insights into GCN and shows its close connection with label propagation. Second, we propose GLP and IGCN to successfully tackle label insufficiency in semi-supervised learning. Third, we demonstrate the high efficacy and efficiency of the proposed methods on various semi-supervised classification and regression tasks.

2. Graph Filtering

This section introduces the concepts of graph signal, graph convolutional filter, and graph filtering.

Notations. A non-oriented graph $\mathcal{G} = (\mathcal{V}, W, X)$ with $n = |\mathcal{V}|$ vertices is given, with a nonnegative symmetric affinity matrix $W \in \mathbb{R}_+^{n\times n}$ encoding edge weights and a feature matrix $X \in \mathbb{R}^{n\times m}$ where an m-dimensional feature vector is associated with each vertex. For semi-supervised classification, only a small subset of vertices are labeled, and the goal is to predict the labels of other vertices. Denote by $Y \in \{0,1\}^{n\times l}$ the label matrix[1], where l is the number of classes.

In graph signal processing [44], the eigenvalues and eigenvectors of the graph Laplacian correspond to frequencies and Fourier basis in classical harmonic analysis. The graph Laplacian is defined as $L = D - W$, where D is the degree matrix. It can be eigen-decomposed as $L = \Phi\Lambda\Phi^{-1}$, where $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_n)$ and $(\lambda_i)_{1\le i\le n}$ are the eigenvalues in the increasing order, and $\Phi = (\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_n)$ and $(\boldsymbol{\phi}_i)_{1\le i\le n}$ are the associated orthogonal eigenvectors. Note that the normalized graph Laplacian $L_r = D^{-1}L$ and the symmetrically normalized graph Laplacian $L_s = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ have similar eigen-decomposition as L. The eigenvalues $(\lambda_i)_{1\le i\le n}$ can be considered as frequencies, and the associated eigenvectors $(\boldsymbol{\phi}_i)_{1\le i\le n}$ form the Fourier basis.

Definition 1 (Graph Signal)**.** A *graph signal* is a real-valued function $f : \mathcal{V} \to \mathbb{R}$ on the vertex set $\mathcal{V}$ of a graph, which can be represented as $\boldsymbol{f} = (f(v_1), \cdots, f(v_n))^\top$ in the vector form.

Any graph signal $\boldsymbol{f}$ can be decomposed into a linear combination of the basis signals $(\phi_i)_{1\le i\le n}$:

$$\boldsymbol{f} = \Phi\boldsymbol{c} = \sum_i c_i\boldsymbol{\phi}_i, \tag{1}$$

where $\boldsymbol{c} = (c_1, \cdots, c_n)^\top$ and c_i is the coefficient of $\boldsymbol{\phi}_i$. The magnitude of the coefficient $|c_i|$ represents the strength

[1] If the label of vertex v_i is known, then $Y(i,:)$ is a one-hot embedding of v_i with $y_{ij} = 1$ if v_i belongs to the j-th class and $y_{ij} = 0$ otherwise. If the label of vertex v_i is not given, then $Y(i,:)$ is a vector of all zeros.

of the basis signal $\boldsymbol{\phi}_i$ presented in the signal $\boldsymbol{f}$. It is well known that the basis signals associated with lower frequencies (smaller eigenvalues) are smoother [64] on the graph, where the smoothness of the basis signal $\boldsymbol{\phi}_i$ is measured by the eigenvalue λ_i, i.e.,

$$\sum_{(v_j,v_k)\in\mathcal{E}} w_{jk}[\boldsymbol{\phi}_i(j)-\boldsymbol{\phi}_i(k)]^2 = {\boldsymbol{\phi}_i}^\top L\boldsymbol{\phi}_i = \lambda_i. \quad (2)$$

Hence, a smooth graph signal $\boldsymbol{f}$ should mostly consist of low-frequency basis signals.

The basic idea of graph filtering is to use the underlying data relation graph to design proper graph filters to produce smooth signals for downstream tasks. A graph filter is a function that takes a graph signal as input and outputs a new signal. A linear graph filter can be represented as a matrix $G \in \mathbb{R}^{n\times n}$, and the output signal is $G\boldsymbol{f}$. In this paper, we focus on graph convolutional filters due to their linear shift-invariant property [42].

Definition 2 (Graph Convolutional Filter)**.** A linear graph filter G is convolutional if and only if there exists a function $p(\cdot) : \mathbb{R} \to \mathbb{R}$, satisfying $G = \Phi p(\Lambda)\Phi^{-1}$, where $p(\Lambda) = \text{diag}(p(\lambda_1), \cdots, p(\lambda_n))$.

The function $p(\cdot)$ is known as the *frequency response function* of the filter G. We shall denote by $p(L)$ the filter with frequency response function $p(\cdot)$.

To produce a smooth signal, the filter G should be able to preserve the low-frequency basis signals in $\boldsymbol{f}$ while filtering out the high-frequency ones. By (1), the output signal can be written as

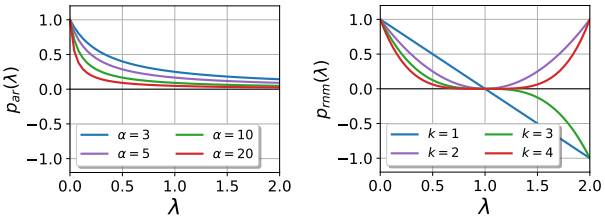
$$\bar{\boldsymbol{f}} = G\boldsymbol{f} = \Phi p(\Lambda)\Phi^{-1}\cdot\Phi\boldsymbol{c} = \sum_i p(\lambda_i)c_i\boldsymbol{\phi}_i. \quad (3)$$

In the output signal $\bar{\boldsymbol{f}}$, the coefficient c_i of the basis signal $\boldsymbol{\phi}_i$ is scaled by $p(\lambda_i)$. To preserve the low-frequency signals and remove the high-frequency ones, $p(\lambda_i)$ should be large for small λ_i and small for large λ_i. Simply put, $p(\cdot)$ should behave like a *low-pass* filter in classical harmonic analysis. Fig. 1(a) shows an example of a low-pass function whose response decreases as the frequency increases.

Taking the vertex features as graph signals, e.g., a column of the feature matrix X can be considered as a graph signal, graph filtering provides a principled way to integrate graph structures and vertex features for learning. In the following, we will revisit two popular semi-supervised learning methods – label propagation and graph convolutional networks under this framework and gain new insights for improving their modelling capabilities.

3. Revisit and Extend Label Propagation

Label propagation (LP) [63, 61, 5] is arguably the most popular method for graph-based semi-supervised learning.



(a) $p_{\text{ar}}(\lambda) = (1+\alpha\lambda)^{-1}$ (b) $p_{\text{rmm}}(\lambda) = (1-\lambda)^k$

Figure 1: Frequency response functions.

As a simple and effective tool, it has been widely used in many scientific research fields and numerous industrial applications. The objective of LP is to find a prediction (embedding) matrix $Z \in \mathbb{R}^{n\times l}$ that agrees with the label matrix Y while being smooth on the graph such that nearby vertices have similar embeddings:

$$Z = \arg\min_Z \{ \underbrace{||Z-Y||_2^2}_{\text{Least square fitting}} + \underbrace{\alpha\text{Tr}(Z^\top LZ)}_{\text{Laplcacian regularization}} \}, \quad (4)$$

where α is a balancing parameter controlling the degree of Laplacian regularization. In (4), the fitting term enforces the prediction matrix Z to agree with the label matrix Y, while the regularization term makes each column of Z smooth along the graph edges. A closed-form solution of the above unconstrained quadratic optimization can be obtained by taking the derivative of the objective function and setting it to zero:

$$Z = (I+\alpha L)^{-1}Y. \quad (5)$$

Each unlabeled vertex v_i is then classified by simply comparing the elements in $Z(i,:)$ or with some normalization applied on the columns of Z first [63].

3.1. Revisit Label Propagation

From the perspective of graph filtering, we show that LP is comprised of three components: signal, filter, and classifier. We can see from (5) that the input signal matrix of LP is simply the label matrix Y, where each column $Y(:,i)$ can be considered as a graph *signal*. Note that in $Y(:,i)$, only the labeled vertices in class i have value 1 and others 0.

The graph *filter* of LP is the Auto-Regressive (AR) filter [48]:

$$p_{\text{ar}}(L) = (I+\alpha L)^{-1} = \Phi(I+\alpha\Lambda)^{-1}\Phi^{-1}, \quad (6)$$

with the frequency response function:

$$p_{\text{ar}}(\lambda_i) = \frac{1}{1+\alpha\lambda_i}. \quad (7)$$

Note that this also holds for the normalized graph Laplacians. As shown in Fig. 1(a), $p_{\text{ar}}(\lambda_i)$ is low-pass. For any

$\alpha > 0$, $p_{\text{ar}}(\lambda_i)$ is near 1 when λ_i is close to 0, and $p_{\text{ar}}(\lambda_i)$ decreases and approaches 0 as λ_i increases. Applying the AR filter on the signal $Y(:, i)$ will produce a smooth signal $Z(:, i)$, where vertices of the same class have similar values and those of class i have larger values than others under the cluster assumption. The parameter α controls the strength of the AR filter. When α increases, the filter becomes more low-pass (Fig. 1(a)) and will produce smoother signals.

Finally, LP adopts a nonparametric *classifier* on the embeddings to classify the unlabeled vertices, i.e., the label of an unlabeled vertex v_i is given by $y_i = \arg\max_j Z(i, j)$.

3.2. Generalized Label Propagation Methods

The above analysis shows that LP only takes into account the given graph W and the label matrix Y, but without using the feature matrix X. This is one of its major limitations in dealing with datasets that provide both W and X, e.g., citation networks. Here, we propose generalized label propagation (GLP) methods by naturally extending the three components of LP.

- Signal: Use the feature matrix X instead of the label matrix Y as input signals.
- Filter: The filter G can be any low-pass graph convolutional filter.
- Classifier: The classifier can be any classifer trained on the embeddings of labeled vertices.

GLP consists of two simple steps. First, a low-pass filter G is applied on the feature matrix X to obtain a smooth feature matrix $\bar{X} \in \mathbb{R}^{n \times m}$:

$$\bar{X} = GX. \tag{8}$$

Second, a supervised classifier (e.g., multilayer perceptron, convolutional neural networks, support vector machines, etc.) is trained with the filtered features of labeled vertices, which is then applied on the filtered features of unlabeled vertices to predict their labels.

GLP has the following advantages. First, by injecting graph relations into data features, it can produce more useful data representations for the downstream classification task. Second, it offers the flexibility of using computationally efficient filters and conveniently adjusting their strength for different application scenarios. Third, it allows taking advantage of powerful domain-specific classifiers for high-dimensional data features, e.g., a multilayer perceptron for text data and a convolutional neural network for image data.

4. Revisit and Improve Graph Convolutional Networks

The recently proposed graph convolutional networks (GCN) [32] has demonstrated superior performance in semi-supervised learning and attracted much attention. The GCN model consists of three steps. First, a so-called renormalization trick is applied on the adjacency matrix W by adding an self-loop to each vertex, resulting in a new adjacency matrix $\tilde{W} = W + I$ with the degree matrix $\tilde{D} = D + I$, which is then symmetrically normalized as $\tilde{W}_s = \tilde{D}^{-\frac{1}{2}} \tilde{W} \tilde{D}^{-\frac{1}{2}}$. Second, define the layerwise propagation rule:

$$H^{(t+1)} = \sigma\left(\tilde{W}_s H^{(t)} \Theta^{(t)}\right), \tag{9}$$

where $H^{(t)}$ is the matrix of activations fed to the t-th layer and $H^{(0)} = X$, $\Theta^{(t)}$ is the trainable weight matrix in the layer, and σ is the activation function, e.g., $\text{ReLU}(\cdot) = \max(0, \cdot)$. The graph convolution is defined as multiplying the input of each layer with the renormalized adjacency matrix $\tilde{W}_s$ from the left, i.e., $\tilde{W}_s H^{(t)}$. The convoluted features are then fed into a projection matrix $\Theta^{(t)}$. Third, stack two layers up and apply a softmax function on the output features to produce a prediction matrix:

$$Z = \text{softmax}\left(\tilde{W}_s \, \text{ReLU}\left(\tilde{W}_s X \Theta^{(0)}\right) \Theta^{(1)}\right), \tag{10}$$

and then train the model with the cross-entropy loss on labeled samples.

4.1. Revisit Graph Convolutional Networks

In this section, we interpret GCN under the graph filtering framework and explain its implicit design features including the choice of the normalized graph Laplacian and the renormalization trick on the adjacency matrix.

GCN conducts graph filtering in each layer with the filter $\tilde{W}_s$ and the signal matrix $H^{(t)}$. We have $\tilde{W}_s = I - \tilde{L}_s$, where $\tilde{L}_s$ is the symmetrically normalized graph Laplacian of the graph $\tilde{W}$. Eigen-decompose $\tilde{L}_s$ as $\tilde{L}_s = \Phi \tilde{\Lambda} \Phi^{-1}$, then the filter is

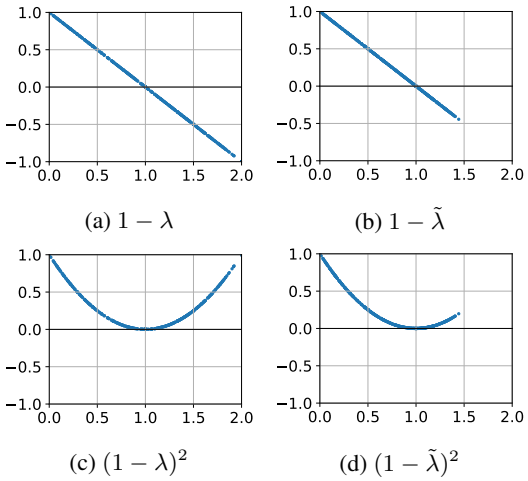
$$\tilde{W}_s = I - \tilde{L}_s = \Phi(I - \tilde{\Lambda})\Phi^{-1}, \tag{11}$$

with frequency response function

$$p(\tilde{\lambda}_i) = 1 - \tilde{\lambda}_i. \tag{12}$$

Clearly, as shown in Fig. 1(b), this function is linear and low-pass on [0, 1], but not on [1, 2].

It can be seen that by performing all the graph convolutions in (10) first, i.e., by exchanging the renormalized adjacency matrix $\tilde{W}_s$ in the second layer with the internal ReLU function, GCN is a special case of GLP, where the input signal matrix is X, the filter is $\tilde{W}_s^2$, and the classifier is a two-layer multi-layer perceptron (MLP). One can also see that GCN stacks two convolutional layers because $\tilde{W}_s^2$ is more low-pass than $\tilde{W}_s$, which can be seen from Fig. 1(b) that $(1 - \lambda)^2$ is sort of more low-pass than $(1 - \lambda)$ by suppressing the large eigenvalues harder .



(a) $1-\lambda$ (b) $1-\tilde{\lambda}$ (c) $(1-\lambda)^2$ (d) $(1-\tilde{\lambda})^2$

Figure 2: Effect of the renormalization trick. Left two figures plot points $(\lambda_i, p(\lambda_i))$. Right two figures plot points $(\tilde{\lambda}_i, p(\tilde{\lambda}_i))$.

Why Use Normalized Graph Laplacian. GCN uses the normalized Laplacian L_s because the eigenvalues of L_s fall into $[0, 2]$ [18], while those of the unnormalized Laplacian L are in $[0, +\infty]$. If using L, the frequency response in (12) will amplify eigenvalues in $[2, +\infty]$, which will introduce noise and undermine performance.

Why the Renormalization Trick Works. We illustrate the effect of the renormalization trick used in GCN in Fig. 2, where the frequency responses on the eigenvalues of L_s and $\tilde{L}_s$ on the Cora citation network are plotted respectively. We can see that by adding a self-loop to each vertex, the range of eigenvalues shrinks from [0, 2] to [0, 1.5], which avoids amplifying eigenvalues near 2 and reduces noise. Hence, although the response function $(1-\lambda)^k$ is not completely low-pass, the renormalization trick shrinks the range of eigenvalues and makes $\tilde{L}_s$ resemble a low-pass filter. It can be proved that if the largest eigenvalue of L_s is λ_m, then all the eigenvalues of $\tilde{L}_s$ are no larger than $\frac{d_m}{d_m+1}\lambda_m$, where d_m is the largest degree of all vertices.

4.2. Improved Graph Convolutional Networks

One notable drawback of the current GCN model is that one cannot easily control filter strength. To increase filter strength and produce smoother features, one has to stack multiple layers. However, since in each layer the convolution is coupled with a projection matrix by the ReLU, stacking many layers will introduce many trainable parameters. This may lead to severe overfitting when label rate is small, or it will require extra labeled data for validation and model selection, both of which are not label efficient.

To fix this, we propose an improved GCN model (IGCN) by replacing the filter $\tilde{W}_s$ with $\tilde{W}_s^k$:

$$Z = \text{softmax}\left(\tilde{W}_s^k \text{ ReLU}\left(\tilde{W}_s^k X\Theta^{(0)}\right)\Theta^{(1)}\right). \quad (13)$$

We call $p_{\text{rnm}}(\tilde{L}_s) = \tilde{W}_s^k$ the renormalization (RNM) filter, with frequency response function

$$p_{\text{rnm}}(\lambda) = \left(I - \tilde{\lambda}\right)^k. \quad (14)$$

IGCN can achieve label efficiency by using the exponent k to conveniently adjust the filter strength. In this way, it can maintain a shallow structure with a reasonable number of trainable parameters to avoid overfitting.

5. Filter Strength and Computation

The strength of the AR and RNM filters is controlled by the parameters α and k respectively. However, choosing appropriate α and k for different application scenarios is non-trivial. An important factor that should be taken into account is label rate. Intuitively, when there are very few labels in each class, one should increase filter strength such that distant nodes can have similar feature representations as the labeled nodes for the ease of classification. However, over-smoothing often results in inaccurate class boundaries. Therefore, when label rate is reasonably large, it would be desirable to reduce filter strength to preserve feature diversity in order to learn more accurate class boundaries.

Fig. 3 visualizes the raw and filtered features of Cora produced by the RNM filter and projected by t-SNE [49]. It can be seen that as k increases, the RNM filter produces smoother embeddings, i.e., the filtered features exhibit a more compact cluster structure, making it possible for classification with few labels.

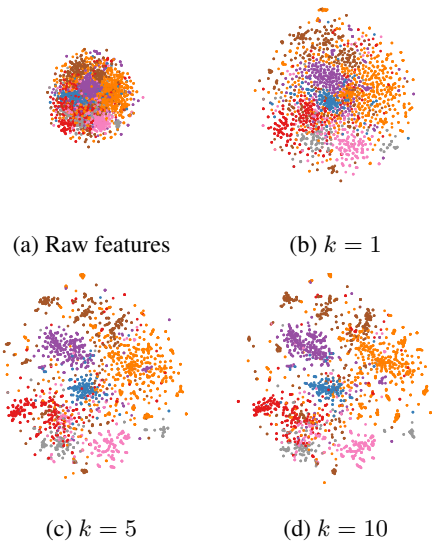
The computation of the AR filter $p_{\text{ar}}(L) = (I + \alpha L)^{-1}$ involves matrix inversion, which is computationally expensive with complexity $\mathcal{O}(n^3)$. Fortunately, we can circumvent this problem by approximating p_{ar} using its polynomial expansion:

$$(I + \alpha L)^{-1} = \frac{1}{1+\alpha}\sum_{i=0}^{+\infty}\left[\frac{\alpha}{1+\alpha}W\right]^i, (\alpha > 0). \quad (15)$$

We can then compute $\bar{X} = p_{\text{ar}}(L)X$ iteratively with

$$X'^{(0)} = \mathbf{O}, \cdots, X'^{(i+1)} = X + \frac{\alpha}{1+\alpha}WX'^{(i)},$$

and let $\bar{X} = \frac{1}{1+\alpha}X'^{(k)}$. Empirically, we find that $k = \lceil 4\alpha \rceil$ is enough to get a good approximation. Hence, the computational complexity is reduced to $\mathcal{O}(nm\alpha + Nm\alpha)$ (note that X is of size $n \times m$), where N is the number of nonzero entries in L, and $N \ll n^2$ when the graph is sparse.



(a) Raw features (b) $k = 1$ (c) $k = 5$ (d) $k = 10$

Figure 3: Visualization of raw and filtered Cora features (by using the RNM filter with different k).

For the RNM filter $p_{\text{rnm}}(\tilde{L}_s) = \tilde{W}_s^k = \left(I - \tilde{L}_s\right)^k$, note that for a sparse graph, $(I - \tilde{L}_s)$ is a sparse matrix. Hence, the fastest way to compute $\bar{X} = p_{\text{rnm}}(\tilde{L}_s)X$ is to left multiply X by $(I - \tilde{L}_s)$ repeatedly for k times, which has the computational complexity $\mathcal{O}(Nmk)$.

6. Experiments

To validate the performance of our methods GLP and IGCN, we conduct experiments on various semi-supervised classification tasks and a semi-supervised regression task for zero-shot image recognition.

6.1. Semi-Supervised Classification

For semi-supervised classification, we test our methods GLP and IGCN on two tasks.[2] 1) Semi-supervised document classification on citation networks, where nodes are documents and edges are citation links. The goal is to classify the type of the documents with only a few labeled documents. 2) Semi-supervised entity classification on a knowledge graph. A bipartite graph is extracted from the knowledge graph [56], and there are two kinds of nodes: entity and relation, where the edges are between the entity and relation nodes. The goal is to classify the entity nodes with only a few labeled entity nodes.

Datasets. We evaluate our methods on four citation networks – Cora, CiteSeer, PubMed [43] and Large Cora, and one knowledge graph – NELL [11]. The dataset statistics are summarized in Table 1. On citation networks, we test two scenarios – 4 labels per class and 20 labels per class. On NELL, we test three scenarios – 0.1%, 1% and 10% label rates.

[2]Code is available at https://github.com/liqimai/Efficient-SSL

Table 1: Dataset statistics.

Dataset	Vertices	Edges	Classes	Features
Cora	2,708	5,429	7	1433
CiteSeer	3,327	4,732	6	3703
PubMed	19,717	44,338	3	500
Large Cora	11,881	64,898	10	3780
NELL	65,755	266,144	210	5414

Baselines. We compare GLP and IGCN with the state-of-the-art semi-supervised classification methods: manifold regularization (ManiReg) [4], semi-supervised embedding (SemiEmb) [53], DeepWalk [40], iterative classification algorithm (ICA) [43], Planetoid [56], graph attention networks (GAT) [51], multi-layer perceptron (MLP), LP [54], and GCN [32].

Settings. We use MLP as the classifier of GLP, and test GLP and IGCN with RNM and AR filters. We follow [32] to use a two-layer structure for all neural networks, including MLP, GCN, IGCN. Guided by our analysis in section 5, the filter parameters k and α should be set large if the label rate is small, and should be set small if the label rate is large. Specifically, when 20 labels per class on citation networks are available or 10% entities of NELL are labeled, we set $k = 5$ for RNM and $\alpha = 10$ for AR filters in GLP. In other scenarios with less labels, we set $k = 10, \alpha = 20$ for GLP. The k, α choosen for IGCN is equal to the above k, α divided by the number of layers – 2. We follow [32] to set the parameters of MLP, GCN, IGCN: for citation networks, we use a two-layer network with 16 hidden units, 0.01 learning rate, 0.5 dropout rate, and 5×10^{-4} L2 regularization, except that the hidden layer is enlarged to 64 units for Large Cora; for NELL, we use a two-layer network with 64 hidden units, 0.01 learning rate, 0.1 dropout rate, and 1×10^{-5} L2 regularization. For more fair comparison with different baselines, we do not use a validation set for model selection as in [32], instead we select the model with the lowest training loss in 200 steps. All results are averaged over 50 random splits of the dataset. We set α of LP to 100 for citation networks and 10 for NELL. Parameters of GAT are same as [51]. Results of other baselines are taken from [56, 32].

Performance of GLP and IGCN. The results are summarized in Table 2, where the top 3 classification accuracies are highlighted in bold. Overall, GLP and IGCN perform best. Especially when the label rates are very small, they significantly outperform the baselines. Specifically, on citation networks, with 20 labels per class, GLP and IGCN perform slightly better than GCN and GAT, but outperform other baselines by a considerable margin. With 4 labels per

Table 2: Classification accuracy and running time on citation networks and NELL.

Label rate	20 labels per class				4 labels per class				10%	1%	0.1%
	Cora	CiteSeer	PubMed	Large Cora	Cora	CiteSeer	PubMed	Large Cora	NELL		
ManiReg	59.5	60.1	70.7	-	-	-	-	-	63.4	41.3	21.8
SemiEmb	59.0	59.6	71.7	-	-	-	-	-	65.4	43.8	26.7
DeepWalk	67.2	43.2	65.3	-	-	-	-	-	79.5	72.5	58.1
ICA	75.1	**69.1**	73.9	-	62.2	49.6	57.4	-	-	-	-
Planetoid	75.7	64.7	77.2	-	43.2	47.8	64.0	-	84.5	75.7	61.9
GAT	79.5	68.2	76.2	67.4	66.6	55.0	64.6	46.4	-	-	-
MLP	55.1 (0.6s)	55.4 (0.6s)	69.5 (0.6s)	48.0 (0.8s)	36.4 (0.6s)	38.0 (0.5s)	57.0 (0.6s)	30.8 (0.6s)	63.6 (2.1s)	41.6 (1.1s)	16.7 (1.0s)
LP	68.8 (0.1s)	48.0 (0.1s)	72.6 (0.1s)	52.5 (0.1s)	56.6 (0.1s)	39.5 (0.1s)	61.0 (0.1s)	37.0 (0.1s)	84.5 (0.7s)	75.1 (1.8s)	**65.9** (1.9s)
GCN	79.9 (1.3s)	68.6 (1.7s)	**77.6** (9.6s)	67.7 (7.5s)	65.2 (1.3s)	55.5 (1.7s)	67.7 (9.8s)	48.3 (7.4s)	81.6 (33.5s)	63.9 (33.5s)	40.7 (33.2s)
IGCN(RNM)	**80.9** (1.2s)	69.0 (1.7s)	77.3 (10.0s)	**68.9** (7.9s)	**70.3** (1.3s)	**57.4** (1.7s)	**69.3** (10.3s)	**52.1** (8.1s)	**85.9** (42.4s)	**76.7** (44.0s)	**66.0** (46.6s)
IGCN(AR)	**81.1** (2.2s)	**69.3** (2.6s)	**78.2** (11.9s)	**69.2** (11.0s)	**70.3** (3.0s)	**58.0** (3.4s)	**70.1** (13.6s)	**52.5** (13.6s)	**85.4** (77.9s)	**75.7** (116.0s)	**67.4** (116.0s)
GLP(RNM)	80.3 (0.9s)	68.8 (1.0s)	77.1 (0.6s)	68.4 (1.8s)	**68.0** (0.7s)	56.7 (0.8s)	68.7 (0.6s)	51.1 (1.1s)	**86.0** (35.9s)	**76.1** (37.3s)	65.4 (38.5s)
GLP(AR)	**80.8** (1.0s)	**69.3** (1.2s)	**78.1** (0.7s)	**69.0** (2.4s)	67.5 (0.8s)	**57.3** (1.1s)	**69.7** (0.8s)	**51.6** (2.3s)	80.3 (57.4s)	67.4 (76.6s)	55.2 (78.6s)

Table 3: Results for unseen classes in AWA2.

Method	Devise	SYNC	GCNZ	GPM	DGPM	ADGPM	IGCN(RNM)			GLP(RNM)		
							k=1	k=2	k=3	k=2	k=4	k=6
Accuracy	59.7	46.6	68.0 (1840s)	**77.3** (864s)	67.2 (932s)	76.0 (3527s)	**77.9** (864s)	**77.7** (1583s)	73.1 (2122s)	76.0 (12s)	75.0 (13s)	73.0 (11s)

class, GLP and IGCN significantly outperform all the baselines, demonstrating their label efficiency. On NELL, GLP and IGCN with the RNM filter as well as IGCN with the AR filter slightly outperform two very strong baselines – LP and Planetoid, and outperform other baselines by a large margin.

Table 2 also reports the running time of the methods tested by us. We can see that GLP with the RNM filter runs much faster than GCN on most cases, and IGCN with the RNM filter has similar time efficiency as GCN.

Results Analysis. Compared with methods that only use graph information, e.g., LP and DeepWalk, the large performance gains of GLP and IGCN clearly come from leveraging both graph and feature information. Compared with methods that use both graph and feature information, e.g., GCN and GAT, GLP and IGCN are much more label efficient. The reason is that they allow using stronger filters to extract higher level data representations to improve performance when label rates are low, which can be easily achieved by increasing the filter parameters k and α. But this cannot be easily achieved in the original GCN. As explained in section 4, to increase smoothness, GCN needs to stack many layers, but a deep GCN is difficult to train with few labels.

6.2. Semi-Supervised Regression

The proposed GLP and IGCN methods can also be used for semi-supervised regression. In [52], GCN was used for zero-shot image recognition with a regression loss. Here, we replace the GCN model used in [52] with GLP and IGCN to test their performance on the zero-shot image recognition task.

Zero-shot image recognition in [52] is to learn a visual classifier for the categories with zero training examples, with only text descriptions of categories and relationships between categories. In particular, given a pre-trained CNN for known categories, [52] proposes to use a GCN to learn the model/classifier weights of unseen categories in the last-layer of the CNN. It first takes the word embedding of each category and the relations among all the categories (WordNet knowledge graph) as the inputs of GCN, then trains the GCN with the model weights of known categories in the last-layer of the CNN, and finally predicts the model weights of unseen categories.

Datasets. We evaluate our methods and baselines on the ImageNet [41] benchmark. ImageNet is an image database organized according to the WordNet hierarchy. All categories of ImageNet form a graph through "is a kind of" relation. For example, drawbridge is a kind of bridge, bridge is a kind of construction, and construction is a kind of artifact. According to [52], the word embedding of each category is learned from Wikipedia by the GloVe text model [39].

Baselines. We compare our methods GLP and IGCN with six state-of-the-art zero-shot image recognition methods, namely Devise [22], SYNC [12], GCNZ [52], GPM [29], DGPM [29] and ADGPM [29]. The prediction accuracy of

these baselines are taken from their papers. Notably, the GPM model is exactly our IGCN with $k = 1$.

Settings. There are 21K different classes in ImageNet. We split them into a training set and a test set similarly as in [29]. A ResNet-50 model was pre-trained on the ImageNet 2012 with 1k classes. The weights of these 1000 classes in the last layer of CNN are used to train GLP and IGCN for predicting the weights of the remaining classes. The evaluation of zero-shot image recognition is conducted on the AWA2 dataset [55], which is a subset of ImageNet. For IGCN and the classifier (MLP) of GLP, we both use a two-layer structure with 2048 hidden units. We test $k = 1, 2, 3$ for IGCN and $k = 2, 4, 6$ for GLP. Results are averaged over 20 runs.

Performance and Results Analysis. The results are summarized in Table 3, where the top 3 classification accuracies are highlighted in bold. We can see that IGCN with $k = 1, 2$ and GPM [29] perform the best, and outperform other baselines including Devise [22], SYNC [12], GCNZ [52] and DGPM [29] by a significant margin. GLP with $k = 2$ is the second best compared with the baselines, only slightly lower than GPM. We observe that smaller k achieves better performance on this task, which is probably because the diversity of features (classifier weights) should be preserved for the regression task [29]. This also explains why DGPM [29] (that expands the node neighborhood by adding distant nodes) does not perform very well. It is also worth noting that by replacing the 6-layer GCN in GCNZ with a 2-layer IGCN with $k = 3$ and a GLP with $k = 6$, the performance boosts from 68% to around 73%, demonstrating the low complexity and training efficiency of our methods. Another thing to notice is that GLP runs hundreds of times faster than GCNZ, and tens of times faster than others.

7. Related Works

There is a vast literature on semi-supervised learning [13, 64], including generative models [2, 31], semi-supervised support vector machine [6], self-training [24], co-training [9], and graph-based methods [30, 34, 35, 59].

Early graph-based methods adopt a common assumption that nearby vertices are likely to have same labels. One approach is to learn smooth low-dimensional embeddings with Markov random walks [45], Laplacian eigenmaps [3], spectral kernels [14, 57], and context-based methods [40]. Another line of works rely on graph partition, where the cuts should agree with the labeled vertices and be placed in low density regions [7, 8, 28, 63], among which the most popular one is perhaps the label propagation methods [5, 15, 61]. It was shown in [21, 23] that they can be interpreted as low-pass graph filtering. To further improve learning performance, many methods were proposed to jointly model graph structures and data features. Iterative classification algorithm [43] iteratively classifies an unlabeled data by using its neighbors' labels and features. Manifold regularization [4], deep semi-supervised embedding [53], and Planetoid [56] regularize a supervised classifier with a Laplacian regularizer or an embedding-based regularizer.

Inspired by the success of convolutional neural networks (CNN) on grid-structured data such as image and video, a series of works proposed a variety of graph convolutional neural networks [10, 27, 20, 1] to extend CNN to general graph-structured data. To avoid the expensive eigen-decomposition, ChebyNet [19] uses a polynomial filter represented by k-th order polynomials of graph Laplacian via Chebyshev expansion. Graph convolutional networks (GCN) [32] further simplifies ChebyNet by using a localized first-order approximation of spectral graph convolution, and has achieved promising results in semi-supervised learning. It was shown in [33] that the success of GCN is due to performing Laplacian smoothing on data features. MoNet [38] shows that various non-Euclidean CNN methods including GCN are its particular instances. Other related works include GraphSAGE [25], graph attention networks [51], attention-based graph neural network [47], graph partition neural networks [36], FastGCN [16], dual graph convolutional neural network [65], stochastic GCN [17], Bayesian GCN [58], deep graph infomax [50], LanczosNet [37], etc. We refer readers to two comprehensive surveys [60, 62] for more discussions.

Another related line of research is feature smoothing, which has long been used in computer graphics for fairing 3D surface [46]. [26] proposed manifold denoising (MD) by using feature smoothing as a preprocessing step for semi-supervised learning, where the denoised data features are used to construct a graph for running a label propagation algorithm. MD uses the data features to construct a graph and employs the AR filter for feature smoothing. However, it cannot be directly applied to datasets such as citation networks where the graph is given.

8. Conclusion

This paper studies semi-supervised learning from a unifying graph filtering perspective, which offers new insights into the classical label propagation methods and the recently popular graph convolution networks. Based on the analysis, we propose generalized label propagation methods and improved graph convolutional networks to extend their modeling capabilities and achieve label efficiency. In the future, we plan to investigate the design and automatic selection of proper graph filters for various application scenarios and apply the proposed methods to solve more real applications.

Acknowledgments

This research was supported by the grants 1-ZVJJ and G-YBXV funded by the Hong Kong Polytechnic University.

References

[1] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In *Conference on Neural Information Processing Systems*, pages 1993–2001, 2016. 8

[2] S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Conference on Neural Information Processing Systems*, pages 854–860, 1998. 8

[3] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(1):209–239, 2004. 8

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(1):2399–2434, 2006. 1, 6, 8

[5] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. *Semi-supervised Learning*, pages 193–216, 2006. 3, 8

[6] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Conference on Neural Information Processing Systems*, pages 368–374, 1998. 8

[7] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning*, pages 19–26, 2001. 8

[8] A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *International Conference on Machine Learning*, page 13, 2004. 8

[9] A. Blum and T. M. Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Computational Learning Theory*, pages 92–100, 1998. 8

[10] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*, 2014. 8

[11] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence*, pages 1306–1313, 2010. 6

[12] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016. 7, 8

[13] O. Chapelle, B. Schölkopf, A. Zien, et al. *Semi-supervised Learning*. MIT Press, 2006. 1, 8

[14] O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. In *Conference on Neural Information Processing Systems*, pages 601–608, 2003. 8

[15] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005. 8

[16] J. Chen, T. Ma, and C. Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018. 8

[17] J. Chen, J. Zhu, and L. Song. Stochastic training of graph convolutional networks with variance reduction. In *International Conference on Machine Learning*, pages 941–949, 2018. 8

[18] F. R. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997. 5

[19] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Conference on Neural Information Processing Systems*, pages 3844–3852, 2016. 8

[20] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Conference on Neural Information Processing Systems*, pages 2224–2232, 2015. 8

[21] V. N. Ekambaram, G. Fanti, B. Ayazifar, and K. Ramchandran. Wavelet-regularized graph semi-supervised learning. In *Global Conference on Signal and Information Processing*, pages 423–426, 2013. 8

[22] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Conference on Neural Information Processing Systems*, pages 2121–2129, 2013. 7, 8

[23] B. Girault, P. Gonçalves, E. Fleury, and A. S. Mor. Semi-supervised learning for graph to signal mapping: A graph signal wiener filter interpretation. In *Conference on Acoustics, Speech and Signal Processing*, pages 1115–1119, 2014. 8

[24] G. Haffari and A. Sarkar. Analysis of semi-supervised learning with the yarowsky algorithm. In *Conference on Uncertainty in Artificial Intelligence*, pages 159–166, 2007. 8

[25] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems*, pages 1024–1034, 2017. 8

[26] M. Hein and M. Maier. Manifold denoising. In *Conference on Neural Information Processing Systems*, pages 561–568, 2007. 8

[27] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 8

[28] T. Joachims. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, pages 290–297, 2003. 8

[29] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing. Rethinking knowledge graph propagation for zero-shot learning. *arXiv preprint arXiv:1805.11724*, 2018. 7, 8

[30] A. Kapoor, Y. A. Qi, H. Ahn, and R. W. Picard. Hyperparameter and kernel learning for graph based semi-supervised classification. In *Conference on Neural Information Processing Systems*, pages 627–634, 2005. 8

[31] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Conference on Neural Information Processing Systems*, pages 3581–3589, 2014. 1, 8

[32] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 1, 2, 4, 6, 8

[33] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, pages 3538–3545, 2018. 2, 8

[34] Y. Li, S. Wang, and Z. Zhou. Graph quality judgement: A large margin expedition. In *International Joint Conference on Artificial Intelligence*, pages 1725–1731, 2016. 8

[35] D. Liang and Y. Li. Lightweight label propagation for large-scale network data. In *International Joint Conference on Artificial Intelligence*, pages 3421–3427, 2018. 8

[36] R. Liao, M. Brockschmidt, D. Tarlow, A. L. Gaunt, R. Urtasun, and R. Zemel. Graph partition neural networks for semi-supervised classification. *arXiv preprint arXiv:1803.06272*, 2018. 8

[37] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1901.01484*, 2019. 8

[38] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Conference on Computer Vision and Pattern Recognition*, pages 5425–5434, 2017. 8

[39] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014. 7

[40] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014. 6, 8

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 7

[42] A. Sandryhaila and J. M. Moura. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656, 2013. 3

[43] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008. 6, 8

[44] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. 2

[45] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Conference on Neural Information Processing Systems*, pages 945–952, 2002. 8

[46] G. Taubin. Curve and surface smoothing without shrinkage. In *International Conference on Computer Vision*, pages 852–857, 1995. 8

[47] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018. 8

[48] N. Tremblay, P. Gonçalves, and P. Borgnat. Design of graph filters and filterbanks. In *Cooperative and Graph Signal Processing*, pages 299–324. 2018. 3

[49] L. Van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 5

[50] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018. 8

[51] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 6, 8

[52] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Conference on Computer Vision and Pattern Recognition*, pages 6857–6866, 2018. 7, 8

[53] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *International Conference on Machine Learning*, pages 1168–1175, 2008. 1, 6, 8

[54] X. Wu, Z. Li, A. M. So, J. Wright, and S.-f. Chang. Learning with Partially Absorbing Random Walks. In *Conference on Neural Information Processing Systems*, pages 3077–3085, 2012. 6

[55] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 8

[56] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pages 40–48, 2016. 1, 6, 8

[57] T. Zhang and R. Ando. Analysis of spectral kernel design based semi-supervised learning. In *Conference on Neural Information Processing Systems*, pages 1601–1608, 2006. 8

[58] Y. Zhang, S. Pal, M. Coates, and D. Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. *arXiv preprint arXiv:1811.11103*, 2018. 8

[59] Y. Zhang, X. Zhang, X. Yuan, and C. Liu. Large-scale graph-based semi-supervised learning via tree laplacian solver. In *AAAI Conference on Artificial Intelligence*, pages 2344–2350, 2016. 8

[60] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *CoRR*, abs/1812.04202, 2018. 2, 8

[61] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Conference on Neural Information Processing Systems*, pages 321–328, 2004. 2, 3, 8

[62] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. 2, 8

[63] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, pages 912–919, 2003. 3, 8

[64] X. Zhu and A. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009. 1, 3, 8

[65] C. Zhuang and Q. Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *International World Wide Web Conference*, pages 499–508, 2018. 8