Combining Graph Attention Mechanism and PageRank to Learn Graph-level Representations

by

Lai Wei

B.Sc., Simon Fraser University University, 2019

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Lai Wei**
Degree:	**Master of Science**
Thesis title:	**Combining Graph Attention Mechanism and PageRank to Learn Graph-level Representations**
Committee:	**Chair:** Qianping Gu Professor, Computing Science
	Martin Ester Supervisor Professor, Computing Science
	Manolis Savva Committee Member Assistant Professor, Computing Science
	Oliver Schulte Examiner Professor, Computing Science

Abstract

Graph convolutional neural networks (GCNs) have revolutionized the field of representation learning on graph data with non-Euclidean properties. As one of its variants, Graph attention networks (GATs) leverage masked self-attentional layers to specify different weights when aggregating node features over the neighbourhoods of graphs. GATs have achieved the state-of-the-arts results across many benchmarking datasets for the task of node classification. However, this method is insufficient in learning graph-level representations for graph classification,which is another important graph learning task. We propose a novel graph pooling method (namely PagePool) to extend GATs to perform graph classification instead of node classification. This method leverages both PageRank message passing algorithm and the attention coefficients of GATs to propagate and calculate the feature-aware node importance estimates (namely attentional PageRank). The attentional PageRank (attPR) values can then be used to select nodes from graphs to get graph-level representations for graph classification.

Keywords: Graph attention networks; Graph representation learning; Graph pooling; Graph classification

Table of Contents

Declaration of Committee **ii**

Abstract **iii**

Table of Contents **iv**

1 Introduction **1**

2 Applications **3**

2.1 Physics 3

2.2 Chemistry and Biology 3

2.3 Social Networks 5

2.4 Image 6

2.5 Text 6

2.6 Generative Models 7

3 Related Work **8**

3.1 Graph Convolution 8

3.2 Graph Pooling 8

3.3 Graph Classification 9

3.4 Attention Models 9

4 Preliminaries **11**

5 Proposed Method **13**

5.1 Attentional PageRank 13

5.2 PagePool 15

5.3 Implementation Details 16

6 Experiments **18**

6.1 Datasets 18

6.2 Baseline Methods 19

6.3 Experimental Settings 19

6.4 Configurations 20
6.5 Results and Analysis 20

7 Future Research Directions **22**
7.1 Deep Graph Embedding. 22
7.2 Dynamic Graph Embedding. 22
7.3 Scalability of Graph Embedding. 23
7.4 Interpretability of Graph Embedding. 23

8 Conclusion **24**

Bibliography **25**

Appendix A Code **31**

Chapter 1

Introduction

In recent years, graph covolutional neural networks (GCNs) [43, 30, 24, 54] have revolutionized the field of graph representation learning through effectively learned node embeddings, and achieved state-of-the-art results in many graph learning tasks, such as link prediction, node and graph classification. GCNs propagate node feature information globally to the entire graph following the spectral rule based on Weisfeiler-Lehman (WL) sub-tree kernel [44]. As a variant of GCNs, graph attention networks (GATs) [48] calculate the attention coefficients between every pair of nodes connected in each 1st-order neighborhood in the graph using only the node features. GATs address some shortcomings of original GCNs by enabling specifying different weights during the message passing of node information. GATs achieve highly competitive results across node classification benchmark datasets. However, the original GATs are potentially insufficient for graph classification as they were designed to learn the node representations only. To make GATs also predict at the graph level, we need to pool together the node representations in order to learn a representation of the entire graph. This task is often referred as graph pooling. Similar to aggregation operations, graph pooling can be regarded as mapping a set of node embeddings to a single graph embedding. The permutation invariant aggregation operations for node embeddings are usually called set pooling. Set pooling method can be simple reduction functions, such as taking the sum (or mean) of the node embeddings with normalization approaches, as in GraphSAGE-pool approach [20]. One downside of the set pooling methods is that they do not exploit the graph topology at the pooling stage. The other type of graph pooling approaches managing to do this are often called graph coarsening. Graph coarsening approaches are inspired by the pooling methods used in convolutional neural networks (CNNs) [8]. The key idea of graph coarsening is to perform clustering algorithms over node presentations for graph pooling. In practice, these coarsening pooling methods can be difficult to train, for they require the clustering functions to be differentiable, while most clustering algorithms are not. To make graph pooling fully differentiable and allows us learn from the graph topology, [61] proposes a novel SortPooling layer which takes unordered node embeddings learned through graph convolution. Instead of performing set pooling or graph coarsening, it sorts node embeddings learned by GCNs in a consistent order and retains a portion of nodes of the input graph as the graph representation. SortPool can be combined with traditional neural network to learn the graph representation in an end-to-end manner, and it deals with input graphs of various sizes.

Inspired by the work of GATs and SortPool, we develop a selection-based graph pooling method (termed as PagePool) that leverages the attention coefficients learned by the masked self-attentional layer using features of every pair of nodes connected, and propagate their normalized values following the message passing algorithm of PageRank [37] to calculate the attentional PageRank (attPR) for every node of the graph. Unlike the traditional PageRank calculation procedure considers only the graph topology, attPR is feature aware. We then sort the learned node embeddings by their attPR values and select a portion of nodes to pool from them. The proposed PagePool method can be integrated into most of graph neural networks (GNNs) models, and it allows GNNs to learn the graph representation in an end-to-end way. Our contributions can be summarized as follows:

- We propose a novel end-to-end graph pooling approach PagePool for graph representation learning. It extends GATs to perform graph classification instead of node classification.
- We propose a novel way to calculate feature-aware PageRank values for node ranking in graphs with node features.
- We evaluate our method on several benchmarking datasets, and compare it with competitive graph classification baselines to demonstrate its effectiveness.

The rest contents of this thesis are organized as follows: In chapter 2, we introduce the applications of graph neural networks in the real world. In chapter 3, we provide a review of some related work to this thesis. The preliminaries used in this paper and some relevant backgrounds are introduced in chapter 4. Chapter 5 describes the graph pooling method PagePool proposed by us in detail. Experiment settings and evaluation results are presented in chapter 6. In chapter 7, we mention several promising future research directions of graph representation learning. Chapter 8 concludes the entire thesis.

Chapter 2

Applications

Graph neural networks have been explored in a wide range of domains across supervised, semi-supervised, unsupervised and reinforcement learning settings as figure 2.1 shows. In this chapter, we give an introduction to applications of graph neural networks for physics, chemistry and biology, social network, image, text and generative tasks.

2.1 Physics

Human intelligence can be understood by modeling real-world systems. To model a physical system, one needs to model the pair-wise interactions between system objects. By representing the objects as nodes and pair-wise interactions as edges, the system can be viewed as graphs. One example is robotics, where bodies and joints can be seen as nodes and edges. The objective is to predict the next state of the bodies based on the current state of the system and the principles of physics.

With studies and explorations of GNNs, GNN-based reasoning are applied on objects, interactions, and physics effectively. [23] takes the trajectory of objects as input and infers an explicit interaction graph, and learns a dynamic model simultaneously. The interaction graphs are learned from former trajectories, and trajectory predictions are generated from decoding the interaction graphs. To generate the robotic system graph, [42] propose a GNN-based model that learns the policy of stably controlling the system by combining reinforcement learning with GNNs.

2.2 Chemistry and Biology

Molecular Representation Learning. Molecular fingerprints is the most common way of representing molecule structures. A simple fingerprint can be a one-hot vector, where each binary value indicates whether a particular substructure exists or not. Such fingerprints can be used for searching molecules, which is an important step of drug design using computers. Extended-connectivity circular fingerprints (ECFP) [41] are standard tools in computational chemistry. Circular fingerprints are generated from fingerprint vectors by encoding which

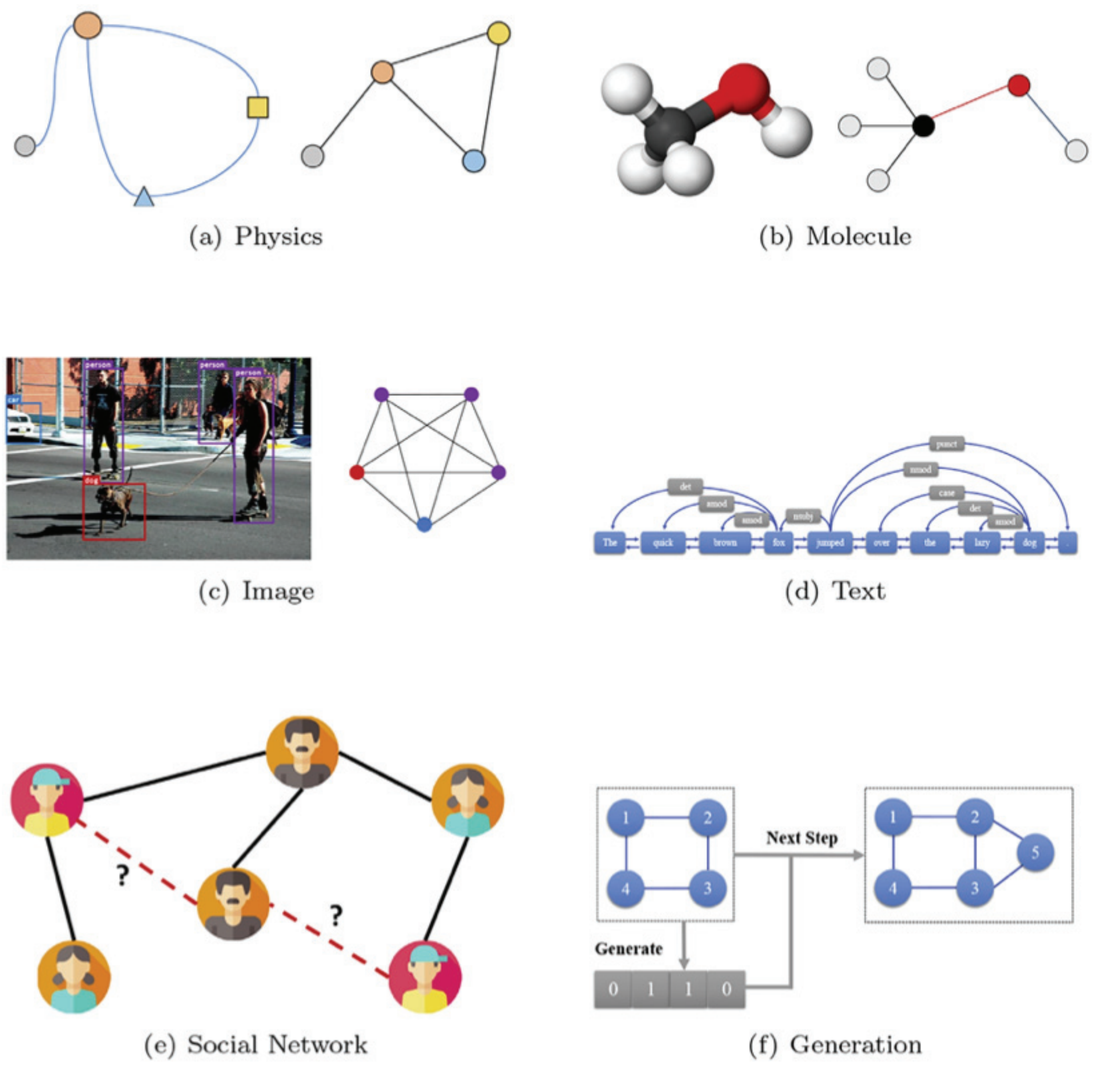

Figure 2.1: Application scenarios of graph neural networks in physics, chemistry and biology, computer vision, social networks, natural language processing, and generation tasks. (Icons made by Freepik from Flaticon)

substructures are present in a molecule in a way that is invariant to atom-relabeling, in which nodes represent atoms, and edges represent bonds. By applying GNNs to molecule graphs like circular fingerprints, we can learn better molecular representations. [10] propose neural graph fingerprints (Neural FPs), which calculate substructure feature vectors via GCNs and sum to get the molecular representations. MolCLR [52] is a self-learning model for molecular representation learning. With three molecular graph augmentation strategies: atom masking, bond deletion and subgraph removal, MolCLR learns informative molecular representation with general GNN backbones.

Chemical Reaction Prediction. In organic chemistry, predicting chemical reaction product is a fundamental problem. [9] propose the Graph Transformation Policy Networks to encode the input molecules and generates an intermediate graph with a node pair prediction network and a policy network.

Protein Interface Prediction. Proteins interact with each other though the interface, which is formed by amino acid residues from each involving protein. Here, the goal is to predict if particular residues constitute part of a protein. Overall, the state of a singe residue depends on its neighboring residues. In the graph of proteins, the residues are represented as nodes. [14] introduces a GCN-based model that learns ligand and receptor protein residue representation and to combine them for making pair-wise classification. [57] invents a way to extract and aggregate local and global node features for making better predictions.

Biomedical Engineering. One popular biomedical engineering task is to model the drug and protein network and deal with different types of edges separately. [40] leverage GCN and relation network to classify breast cancer sub-types. [65] propose a GCN-based model to predict polypharmacy side effects.

2.3 Social Networks

Graph algorithms have been wide used to help solving tasks of social networks. With the emergence of GNNs, problems of social networks can be better explored.

Recommendation Systems. User-item prediction is one of the classic problems for recommendation system. Modeling users and user interactions as nodes and edges in a graph, GNNs can naturally be applied to this problem. [3] propose GC-MC to apply GCNs on user-item rating graphs to learn embeddings of users and items. [4] introduce PinSage to adopt GNNs in web-scale scenarios, which builds computational graphs with weighted sampling strategy for the bipartite graph to reduce repeated computation. Recommendation systems also tries to incorporate user social networks to enhance recommendation performance. GraphRec [13] learns user embeddings from both item side and user side. Going beyond static social activities, [53] model homophily and influence effects by dual attentions.

2.4 Image

Graph neural networks can help solving many image tasks of computer vision that traditional CNNs do not deal very well with.

Reasoning. Reasoning in computer vision is to incorporate both spatial and semantic information of images. Graphs can be generated to perform image reasoning. Visual question answering is a typical reasoning task of computer vision, where the model is given text questions and asked to return the answers about the image. The answers is believed to lie in the spatial relations among objects of the images. [46] builds a question syntactic graph and an image scene graph, then GNN is applied to learning the embeddings and make the predictions. Building a knowledge graph, [36] manage to obtain finer spatial relation and results with better interpretability. GNNs can help other reasoning applications like object detection, interaction detection, and region classification. In object detection, [21] use GNNs to calculate RoI features. In interaction detection [39], GNNs are used to perform message-passing between humans and objects. [5] use GNNs to perform reasoning on graphs that connects regions and classes in region classification.

Semantic Segmentation. The task of semantic segmantatiion is to assign a unique label to each pixel of the image, which can be viewed as a dense classification problem. Traditional CNNs could fail on this task, because image regions are often not grid-like and need to consider global information. GNNs is made to deal with irregular inputs. [31] build graphs in the form of distance-based superpixel map and apply LSTM to propagate neighborhood information globally to capture the long-term dependency with spatial connections. Futher more, 3D semantic segmentation and point clouds classification require more geometric information and therefore are hard to be solved by a 2D CNN. [27] build superpoint graphs and generate graph embeddings to to solve the 3D point cloud segmentation problem. [51] model cloud point interactions through edges, then they compute the edge representation vectors by feeding the coordinates of its terminal nodes. in the end, node embeddings are learned by performing edge aggregation.

2.5 Text

Text tasks can be solved by graph neural networks on both the word-level and sentence-level. Here, we introduce some GNN applications to text. Conventionally, the graph representation of the text can often be achieved by transforming the text into a graph of word nodes, where edges are built among nodes based on word occurrence in documents (document-word edges) and word co-occurrence in the whole corpus (word-word edges).

Text Classification. Text classification is a popular and classical problem in natural language processing. Representing a text as a graph of words can help capturing semantics between non-consecutive and long distance words. [38] utilize a GNN-based model to first transform text into graph-of-words, and then perform graph convolutional operations to convolve the word graph. [58] view words of the text document to construct the corpus graph, then use GCN to learn graph-level representations and classify the text.

Sequence Labeling. The task of sequence labeling is to assign a categorical label for a sequence of observed variables (like words). One typical task is POS-tagging, in which the words in a sentence are labeled by their part-of-speech. Another one is named entity recognition (NER), where to predict if each word in a sentence belongs to a part of a named entity. GNNs can be utilized to address the problem, if each variable in the sequence if viewed as node and their dependencies as edges. [63] propose the sentence LSTM to predict the sequence label. They achieved competitive results on both POS-tagging and NER tasks.

Fact Verification. Fact verification is a task requiring the model to verify given claims based on the evidence extracted from the text. Multiple pieces of evidence need to be reasoned for verifying the claim. [32] propose a GNN-based model KGAT to perform aggregation and reasoning over evidence based on a fully-connected evidence graph. [64] create an inner-sentence graph with the information from semantic role labeling and achieved promising results.

2.6 Generative Models

Generative models are important to real-world graph applications like modeling social interactions, discover chemical structures, and knowledge graph construction.

Some graph generative models generate graphs sequentially. [60] introduce GraphRNN to generate the adjacency matrix of a graph by generating the adjacency vector of each node step by step, which returns graphs with various numbers of nodes. [29] propose a method that generates edges and nodes sequentially and makes use of GNNs to extract the hidden features from the graph to make decision on the actions for the next step in the generation process.

Some other generative models generate the graph adjacency matrix at once rather than doing it sequentially. MolGAN [7] has a permutation-invariant discriminator to solve the node variant problem in the graph adjacency matrix. It also applied a reward network for reinforcement learning optimization towards desired chemical properties. [33] propose constrained variational auto-encoders to ensure the semantic correctness of generated graphs. Graphite [19] integrates GNN into variational auto-encoders to encode the graph structure and features into latent variables. More specifically, it uses isotropic Gaussian as the latent variables and then uses iterative refinement strategy to decode from the latent variables.

Chapter 3

Related Work

In this chapter, we provide a review of some related work to this thesis.

3.1 Graph Convolution

Convolution operation on graphs can be defined in either the spectral or non-spectral domain. Spectral approaches focus on redefining the convolution operation in the Fourier domain, utilizing spectral filters that use the graph Laplacian. Kipf and Welling proposed a layer-wise propagation rule that simplifies the approximation of the graph Laplacian using the Chebyshev expansion method [8]. The goal of non-spectral approaches is to define the convolution operation so that it works directly on graphs. In general non-spectral approaches, the central node aggregates features from adjacent nodes when its features are passed to the next layer rather than defining the convolution operation in the Fourier domain. Hamilton et al. proposed GraphSAGE [20] which learns node embeddings through sampling and aggregation. While GraphSAGE operates in a fixed-size neighborhood, Graph Attention Network (GATs) [48], based on attention mechanisms [2], computes node representations in entire neighborhoods. Both approaches have improved performance on graph-related tasks.

3.2 Graph Pooling

Pooling layers enable CNN models to reduce the number of parameters by scaling down the size of representations, and thus avoid overfitting. To generalize CNNs, the pooling method for GNNs is necessary. Graph pooling methods can be grouped into the following two categories: Global and hierarchical pooling. Global pooling methods use summation or neural networks to pool all the representations of nodes in each layer. Graphs with different structures can be processed because global pooling methods collect all the representations. Gilmer et al. viewed GNNs as message passing schemes, and proposed a general framework [18] for graph classification where entire graph representations could be obtained by utilizing the Set2Set [49] method. SortPool [61] sorts embeddings for nodes according to the structural roles of a graph and feeds the sorted embeddings to the next layers. However, global pooling methods do not learn hierarchical representations which are crucial for capturing structural information of graphs. The main motivation of hierarchical pooling methods is to build a model that can learn feature- or topology-based node assignments in each layer. Ying et al. proposed DiffPool [59] which is a differentiable graph pooling method that can learn assignment matrices in an end-to-end fashion. A learned assignment matrix in layer l, $S(l)$ contains the probability values of nodes in layer l being assigned to clusters in the next layer $l+1$. Here, n^l denotes the number of nodes in layer l. gPool [15] is another hireachical graph pooling method and it achieved performance comparable to that of DiffPool. gPool requires a storage complexity of $O(|V| + |E|)$ whereas DiffPool requires $O(k|V|^2)$ where V, E, and k denote vertices, edges, and pooling ratio, respectively. gPool uses a learnable vector p to calculate projection scores, and then uses the scores to select the top ranked nodes. Projection scores are obtained by the dot product between p and the features of all the nodes. The scores indicate the amount of information of nodes that can be retained.

3.3 Graph Classification

The goal of graph classification is to predict a label for the entire graph, which depends on the representation of the entire graph rather than node embeddings that are often directly obtained from GNNs. [61, 18] tackle the problem via aggregating node representations in a flat, non-hierarchical manner. [45] seek to generate a hierarchical structure through running deterministic graph clustering algorithms on node features. [35] obtains cluster assignments by applying k-means algorithm. However, they are two-stage approaches, and cannot be learned in an end-to-end fashion. [60] propose a sorting strategy to order the nodes whose representations are then concatenated to feed into the final prediction architecture. [15] propose trainable pooling layers that adaptively generate cluster assignments and can be learned in a differentiable way. Graph Isomorphism Network (GIN) [55] is one of the most powerful graph classification model builds upon the limitations of GraphSAGE, extending it with arbitrary aggregation functions on multi-sets. The model is proven to be as theoretically powerful as the Weisfeiler-Lehman test of graph isomorphism.

3.4 Attention Models

Inspired by the success of attention models in deep learning communities [47], graph learning researches have applied attention mechanism to graph neural networks. In recent years, multiple attention-based GNNs have been proposed. They share the same fundamental idea that attentions are calculated to make decision on most task-relevant parts, although they define attentions in different ways and use attentions for various purposes. Attention mechanism benefits GNNs via aggregating node information [48], integrating multiple models [30], or guiding graph random-walk with importance [1]. Here, attention module has two merits in our model: 1. it helps select discriminative nodes to form hierarchical graph structure; 2. it generates the graph representation with attention-weighted pooling. In contrast to simpler sequence-to-sequence models, Graph2Seq [56] is a method that can output a sequence given a general graph. This can be applied to tasks where we are given a graph capturing various relationships between different entities and are asked to answer a text query by reasoning using the provided information. GRAM [6] applies attention to a medical ontology graph to help learning attention-based graph embeddings for medical codes. While the problem they studied is classifying a patient record (described by certain medical codes), where the novelty is the application of attention on the ontology graph to improve model performance.

Chapter 4

Preliminaries

In this chapter, we give an introduction about the preliminaries used in this thesis and some relevant backgrounds.

Let $G = (V, E)$ where G denotes the undirected graph, where V is the set of nodes. In total, the number is $N = |V|$. $M = |E|$ is the set of edges, in total the number is $|E|$. The graph topology of G is represented using an adjacency matrix A, where $A_{i,j} = 1$ if and only if $(i, j) \in E$. G is associated with a node features matrix $H = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}$ of size $N \times F$ and a class label in $\{1, 2, ..., C\}$, where F is the dimension of node features and C is the size of label set.

Graph attentional layer is used by GATs to calculate self-masked attention coefficients and perform graph convolutions. Taking G and H as the inputs, the layer produces a set transformed node features $H' = \{\vec{h'}_1, \vec{h'}_2, ..., \vec{h'}_N\}$ as its output. As an initial step, a shared linear transformation, parameterized by a weight matrix $\mathbf{W}$ of size $F \times F'$ is applied to every node. The self-attention is performed on the pairs of nodes connected: a shared attention mechanism a to compute the attention coefficients

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \tag{4.1}$$

e_{ij} indicates the importance of node $j's$ features to node i. The graph structure is injected into the mechanism by using *mask attention*. Only compute e_{ij} for nodes $j \in N_i$, where N_i is the first-order neighborhood of node i in the graph. To make coefficients comparable across different nodes, the edge softmax function is used for normalization by destination nodes in G:

$$\alpha_{ij} = \textsc{Edge\_sotfmax}_{dst}(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})} \tag{4.2}$$

Once calculated, the normalized attention coefficients are used to compute a linear transformation of the input node features, to serve as the final output features for each of the node. A non-linearity σ is applied after.

$$\vec{h}_i' = \sigma\left(\sum_{j \in N_i} a_{ij} \mathbf{W} \vec{h}_j\right) \tag{4.3}$$

$H' = \{\vec{h'}_1, \vec{h'}_2, ..., \vec{h'}_N\}$ can then be used as the final node representations for node classification or passed as inputs of the next GAT layer to aggregate information from higher-order neighborhoods.

PageRank is a message passing algorithm runs on a graph to measure the importance of nodes in edge connectivity. In each iteration of this algorithm, node i first scatters its PageRank value pr_i uniformly to its downstream nodes. The new pr_i is computed by aggregating the received PageRank values from its neighbors, which is then adjusted by the damping factor d:

$$pr_i = \frac{(1-d)}{N} + d \times \sum_{j \in N_i} \frac{pr_j}{D(j)} \tag{4.4}$$

where $D(j)$ is the out-degree of node j, and N_i are the neighboring nodes of i. pr_i is passed along the directed edges iteratively until it is converged. Once converged, the PageRank values can be viewed as estimates of node importance in the sense of graph topology.

Chapter 5

Proposed Method

In this chapter, we introduce our graph pooling method *PagePool*, and describe how it will be used with graph neural networks to learn graph-level representation for graph classification.

Problem Setting. A graph can be represented as $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node in the graph is associated with node features and we use H of size $N \times F$ to denote the feature matrix, where N is the number of nodes and F is the dimension of node features. The graph structural information is represented by an adjacency matrix A of size $N \times N$. In the graph classification setting, we have a set of graphs $\{G_i\}$, each G_i is associated with a label y_i. The task of graph classification is to take the graph (structural information and node features) as input and predict its corresponding label.

To make the prediction, it is important to extract useful information from both graph structure and node features. We aim to design a graph pooling method PagePool to leverages both PageRank message passing algorithm and the attention coefficients of GATs to propagate and calculate the feature-aware node importance estimates (namely attentional PageRank), which can be used to select a fixed-size node embeddings as the graph-level representation of the input graph for graph classification. In general, we aim to first learn node representations of the input graph and aggregate them into the graph representation that is predictive of its class label.

5.1 Attentional PageRank

PageRank returns a sorting of nodes according to the estimates of their importance in graph topology, but it is not a feature-aware message passing algorithm. For a node i in G, it passes its pagerank value pr_i to neighboring nodes equally by $\frac{pr_i}{D(i)}$ during the message passing phase without considering the feature importance of its neighbors. This is potentially insufficient, because node features are also an important part of the graph. Making PageRank algorithm feature-aware can be critical for finding the most important nodes in a graph. To solve this problem, we propose a way to calculate attentional pagerank values exploiting both graph

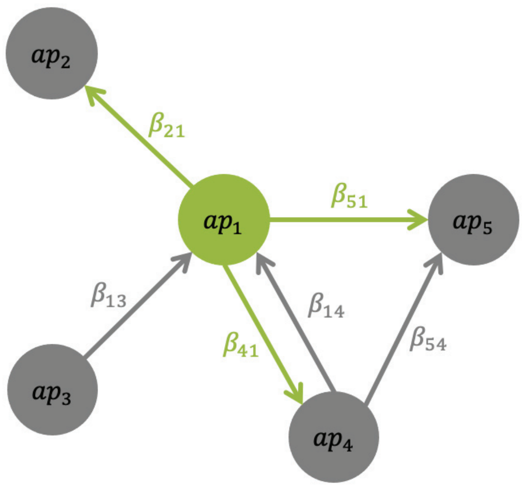

Figure 5.1: One iteration of attentional pagerank message passing. β_{ij} stands for the normalized attention coefficients for the node j of node i. Here, ap_i indicates the attentional pagerank value of node i.

topology and node features. Algorithm 1 provides an overview of the attentional pagerank computation process.

Inspired by graph attentional layers, we first concatenate nodes features of every pair of nodes in G, then calculate the masked self-attention coefficients by

$$e_{ij} = LeakyReLU(\vec{a}^T[\mathbf{W}\vec{h}_i||\mathbf{W}\vec{h}_j]) \tag{5.1}$$

We use a 1-layer multi-layer perceptron as the attention mechanism a, where $||$ stands for feature concatenation. We then normalize e_{ij} values within a 1-hop neighborhood using the edge softmax function by the source nodes.

$$\beta_{ij} = \text{EDGE\_SOTFMAX}_{src}(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in N_j} exp(e_{kj})} \tag{5.2}$$

The normalized attention coefficients β_{ij} for the neighbors of node i can then be used to weight the pagerank values of node i when passing to its neighbors $j \in N_i$. The one iteration of attentional pagerank value ap_i of node i is calculated by

$$ap_i = \frac{(1-d)}{N} + d \times \sum_{j\in N_i} \beta_{ji} \times ap_j \tag{5.3}$$

We repeat the calculation in equation 5.3 until ap_i converges for all node $i \in V$, or the maximum number of iteration n is reached. After the computation of ap_i, we store the attentional pagerank values in vector $attPR$ of size $N \times 1$. Figure 5.1 shows the message passing procedure of attentional pagerank.

Algorithm 1 Attentional pagerank computation algorithm

Input: Graph $G = (V; E)$; Adjacency matrix A; Input node features $H = \{\vec{h}_i, \forall i \in V\}$; Trainable graph attention function $\textsc{Gat}\ (\cdot)$ returns an attention coefficient matrix A', for $A'_{i,j} = e_{ij}$ where $A_{i,j} = 1$; $\textsc{Edge\_softmax}_{src}(\cdot)$ computes softmax over weights of incoming edges by source nodes; Damping factor d; the max number of iterations n

Output: Attentional pagranks $attPR$ for nodes in graph G

```
1:  Initialize $attPR$ to be an array of size $N$ $(N = |V|)$
2:  Initialize each $attPR(i)$ to $1/N$
3:  Initialize $attPR'$ to be an array of $N$ zeros
4:  $A' \leftarrow \textsc{Gat}(\text{A, H})$
5:  $A'_{i,j} \leftarrow \textsc{Edge\_softmax}_{src}(A'_{i,j})$ for $A'_{i,j} \in A'$
6:  $m \leftarrow 0$
7:  while $attPR' \neq attPR$ and $m \leq n$ do
8:      $attPR' \leftarrow attPR$
9:      $m \leftarrow m + 1$
10:     for $i \leftarrow 0$ to $N - 1$ do
11:         for $j \leftarrow 0$ to $N - 1$ do
12:             $attPR(i) \leftarrow attPR(i) + A'_{i,j} \times attPR'(j)$
13:         end for
14:     end for
15: end while
16: return $attPR$
```

5.2 PagePool

One challenge of graph pooling is the need to deal with input graphs of different sizes, more specifically different numbers of nodes, and set pooling is usually not a satisfying solution. In traditional deep learning, CNNs deal with this problem by truncating or padding the input images (or sequences) to the same size. However, this is typically hard to do with graph data because of its non-euclidean properties. SortPool [61] addresses this problem by sorting node embeddings according to their structural roles in the graph, and select a fixed-number of node embeddings to form a graph representation. As attentional pagerank values give us a more accurate estimate of node importance considering both graph topology and node features, inspired by SortPool, we propose the PagePool method that selects a fixed-number of nodes according to their attentional pagerank values to learn the graph-level representation.

PagePool performs graph pooling on graph data in two steps: first, it selects the top-k nodes from the input graph structure according to the attentional pagerank values calculated using masked self-attention coefficients. Second, it performs a global sum pooling on the selected node embeddings to obtain a graph-level representation. The process can be formulated as:

$$
\begin{aligned}
idx &= f_{topk}(attPR, k), \\
H_g &= \sigma\left(\sum_{i \in idx} H'_i\right)
\end{aligned}
\tag{5.4}
$$

$f_{topk}(*)$ is a sorting function and returns indices of k nodes with the largest values of the first argument, where idx is of size $k \times 1$. H'_i indicates the node embeddings learned by GAT of node i, and H_g is the final graph representation of G. H_g is of size $k \times F'$, where F' is the number of node feature dimensions after linear transformation. H_g is then fed to a multi-layer perceptron (MLP) [17] followed by the softmax function for the class label prediction. The classification and loss function are defined as follows

$$
\begin{aligned}
P &= softmax_C(MLP(Z_G)), \\
loss &= -\sum_{i \in C} y_i log(P_i)
\end{aligned}
\tag{5.5}
$$

Here, C is the total number of classes, and $y_i \in Y$ is the true label for graph instance i. For convenience, we call the GAT layers appended by PagePool GAT-PP.

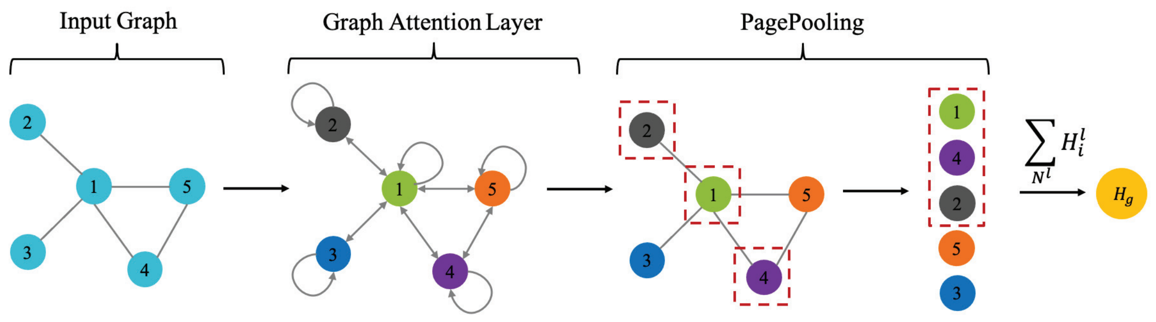

Figure 5.2: Overview of GAT-PP. GAT-PP takes the input graph G, performs aggregation and transformation of node features through calculating the masked self-attentional coefficients. The PagePool layer calculates the attentional pagerank values of each node and sorts their learned node embeddings accordingly. A global sum pooling is performed on the node embeddings of the top-k nodes with the highest attentional pagerank values to obtain H_g, the graph-level representation of G

5.3 Implementation Details

The GAT-PP layers were implemented using Deep Graph Library (DGL) [50], which is a Python package built for easy implementation of various GNN models, on top of existing deep learning frameworks. The details of GAT-PP's python implementation can be found in Appendix A Code.

Attentional PageRank. To compute the attentional pagerank values for the input graph with node features, we leverage the *get_attention* function of the *GATConv* module of DGL to compute and return the unnormalized attention coefficients between every pair of nodes connecting to each other. To normalize the attention coefficients by source nodes and assign weights to edges, we use the DGL function *edge_softmax* setting parameter *norm_by='src'* and add normalized values to the edge features of the graph.

Message Passing. The initial attentional pagerank values are set the same as 5.3. To perform the pagerank message passing for attentional pagerank values, we define a message function *pr_message_func* computes messages only from source node features. We also define the reduce function *pr_reduce_func*, which removes and aggregates the messages from its mailbox, and computes its new pagerank values.

Node Sorting and Selection. After computing the final attentional pagerank values for all the nodes, we sort them in the descending order regarding to the attentional pagerank values and keep only the top-K nodes. This operation can be done by utilizing the DGL function *topk_nodes* setting parameters *k=K, sortby=-1.*

Chapter 6

Experiments

This chapter describes the benchmarking datasets, baseline methods, and details of our experimental settings.

6.1 Datasets

Following the literature on graph representation learning and graph classification [55, 61], we use two bioinformatics datasets and two social network datasets as our benchmarks. All four graph datasets are available in public [34] and frequently used to compare GNN models.

PROTEINS contains the graph representations of protein structures, where nodes represent secondary structure elements and edges represent proximity in 3D space. The graph label indicates whether a protein is classified as enzyme or non-enzyme.

NCI1 is a cheminformatics dataset, where each graph represents a chemical compound: each vertex stands for an atom of the molecule, and edges between vertices represent bonds between atoms. The label of a graph indicates if the chemical is active or inactive against lung cancer cells.

IMDB-BINARY is a movie collaboration dataset that consists of the ego-networks of 1,000 actors/actresses who played roles in movies in IMDB. In each graph, nodes represent actors/actresses, and edges connect those who appear in the same movie. These graphs are derived from the Action and Romance genres, where the graph labels indicate the classes of genres.

Datasets	**PROTEINS**	**NCI1**	**IMDB-B**	**IMDB-M**
# Graphs	1113	4110	1500	1500
Avg. # Nodes	39.1	29.8	19.8	13.0
# Node Features	3	37	1	1
# Classes	2	2	2	3

Table 6.1: Statistics of datasets.

IMDB-MULTI is another movie dataset that consists of a network of 1000 actors or actresses who played roles in movies in IMDB. In a graph, nodes represent actors or actresses, and an edge connects two nodes when they appear in the same movie. These graphs are derived from three genres: Comedy, Romance and Sci-Fiction, where the graph labels indicate the classes of genres.

The statistics of our benchmarking datasets is summarized in table 6.1.

6.2 Baseline Methods

We compare our proposed GAT-PP model with some state-of-the-art GNN methods for graph classification used in the work [11].

- **GraphSAGE**: is the earliest GNN model utilizing the idea of node feature aggregation to learn the graph representation. We choose to use GraphSAGE with mean aggregator as our baseline.
- **DiffPool** is an end-to-end trainable graph coarsening model that produces hierarchical representations of graphs by repeatedly predicting cluster assignments.
- **DGCNN** is a GNN model that learns graph-level representations with SortPool. It provides a way to use GNN modules and traditional deep learning modules together in an end-to-end manner. It is the only baseline model utilizing sort-based pooling strategy to learn graph-level representations.
- **GIN** Graph isomorphism networks (GIN) is a GNN model injects multiset functions for neighbor aggreagation. GIN builds upon the limitations of GraphSAGE and is proven to be as powerful as Weisfeiler-Lehman test.
- **GAT-PR** is a GNN model uses GAT layers for graph convolution and sort nodes by the pre-computed traditional pagerank values for node selection and graph pooling.

6.3 Experimental Settings

All methods were trained and tested in the same way as described in the work [11]. We have also added the node degree as an additional feature to all the datasets, since it was noted as an important way to improve the graph classifiaction performance We use a 10-fold CV for model assessment and an inner holdout technique with a 90%/10% training/validation split for model optimization. After the model optimization, we train models on the whole training fold, holding out a random fraction (10%) of the data to perform early stopping monitoring the cross-entropy loss on validation set. To prevent models from over-fitting, we implement the early stopping strategy with a patience n, where the training will be stopped if the classification loss on the validation set has not been improved for n epoch. We report the average classification accuracy and standard variance across the 10 folds within the cross-validation.

6.4 Configurations

We trained all the methods for 1000 epochs using the Adam optimizer [22] with a learning rate of 0.0001. The damping factor d of PageRank was set to 0.15, which is the commonly used default value [37]. For model optimization, we search though a bunch of hyper-parameter combinations. We list the values that have been tried for each hyper-parameter: (1) the number of hidden units at MLP layer $\in \{64, 128\}$, (2) the batch size $\in \{16, 32\}$, (3) the dropout ratio $\in \{0, 0.5\}$, (4) the node feature embedding dimensions $\in \{32, 64\}$, (5) the number of nodes to be selected from a graph (k), was set such that $\in \{60\%, 70\%\}$ of graphs in a dataset have more than k nodes for bioinformatics datasets, and set $\in \{80\%, 90\%\}$ for scocial network datasets. This is set the same way as used in SortPool [61], (6) the number of GNN layers $\in \{3, 5\}$, (7) the patience for early stopping $\in \{50, 100\}$.

Due to the available computational resources and the amount of experiments to conduct, we limited the time budget for one single training to 72 hours. For all the baseline methods from the literature, we use the results reported by [11] for consistency, as we adopt the same experimental setting.

6.5 Results and Analysis

Table 6.2 reports the graph classification accuracy and standard variances over 10 cross-validation folds. for our proposed method GAT-PP, its simplified version GAT-PR, and five competitive GNN baselines. GAT-PP achieved the best average cross-validation accuracy on PROTEINS and IMDB-B and the second best accuracy on IMDB-M. It is surprising that GAT-PP's performance on NCI1 is not as competitive as expected given NCI1 has the largest number of node features among the four benchmarking datasets. Hypothetically, PagePool should allow GNN models to learn better representations for graphs with rich node features. We suspect that the somewhat disappointing performance is due to the limitations of the GAT module. For instance, as [23] pointed out, the induced attention functions of GATs are prone to over-fitting due to the increasing number of parameters and the lack of direct supervision on attention weights. GATs also suffer from over-smoothing at the decision boundary of nodes. We suppose that combing PagePool with a more advanced version of GATs will lead to an improved performance. We also observe that GAT-PR performs reasonably well compared to the baselines, although it uses merely the traditional pagerank values for graph pooling, which shows the effectiveness of graph pooling done by node ranking. Compared to GAT-PR, GAT-PP obtains a significant accuracy improvement by considering the node features when performing pagerank message passing, which

Datasets	**PROTEINS**	**NCI1**	**IMDB-B**	**IMDB-M**
GraphSage	73.0 ± 4.5	76.0 ± 1.8	68.8 ± 4.5	47.6 ± 3.5
DGCNN	72.9 ± 3.5	76.4 ± 1.7	69.2 ± 3.0	45.6 ± 3.4
DiffPool	73.7 ± 3.5	76.9 ± 1.9	68.4 ± 3.3	45.6 ± 3.4
ECC	72.3 ± 3.4	76.2 ± 1.4	67.7 ± 2.8	43.5 ± 3.1
GIN	73.3 ± 4.0	$\mathbf{80.0} \pm 1.4$	71.2 ± 3.9	$\mathbf{48.5} \pm 3.3$
GAT-PR	73.0 ± 4.8	73.1 ± 1.5	70.4 ± 4.0	46.9 ± 4.1
GAT-PP (ours)	$\mathbf{74.5} \pm 3.6$	76.4 ± 2.4	$\mathbf{72.8} \pm 2.5$	47.8 ± 3.5

Table 6.2: Summary of results in terms of 10-fold classification accuracy.

demonstrates the benefits of the attentional pagerank (attPR). DGCNN is the only baseline model utilizing sort-based pooling. As results highlighted in table 6.2, GAT-PP out performs DGCNN on PROTEINS, IMDB-B and IMDB-M, and achieves about the same performance on NCI1, which demonstrates the effectiveness of our node sorting methods based on attentional pagerank calculation.

We notice that GIN also performs decently on our benchmarks, especially on NCI1, where it outperforms GAT-PP by 3.6% in terms of average classification accuracy. It makes applying the PagePool idea to GIN layers an interesting future research direction too. However, computational complexity may be a challenge, since GIN is not an attention-based model like GAT. We have a limited the number of benchmarking datasets due to the constraints of time and computational resources. To further increase the generality of our results, the experiments could be repeated on the TUdatasets [34]. We also noticed that the classification results in the last column are significantly worse than in the second last column for all models. The classification accuracies diminish vastly because performing three-class classification is much harder than the task of binary classification.

Chapter 7

Future Research Directions

Graph representation learning is a well-motivated topic. It is effective to convert graph data into a low dimensional space in which important feature information is well preserved. Graph analytic can provide researchers with a deeper understanding of the data with the help of efficient graph embedding techniques. There are several future research directions of graph representation learning which is worth mentioning.

7.1 Deep Graph Embedding.

GCNs and some of their adaptations have drawn great attention due to their superior performance. However, the number of graph convolutional layers is typically not greater than two. When there are more graph convolutional layers in cascade, the performance drops significantly. It was argued in [28] that each GCN layer corresponds to graph Laplacian smoothing since node features are propagated in the spectral domain. When a GCNs is deeper, the graph Laplacian is over-smoothed and the corresponding node features become obscure. Each layer of GCN usually only learns one-hop information, and two GCN layers learn the first- and second- order proximity in the graph. It is difficult for a shallow structure to learn global information. One solution to fix this problem is to conduct the convolution in the spatial domain. For example, one can convert graph data into grid-structure data as proposed in [16]. Then, the graph representation can be learned using multiple CNN layers.

7.2 Dynamic Graph Embedding.

Social graphs, such as graphs in Twitter, are always changing. Another example is graphs of mobile users whose location information is changing along with time. To learn the representation of dynamic graphs is an important research topic and it finds applications in real- time and interactive processes such as the optimal travel path planning in a city at traffic hours. Hyper-graphs is a good option for modeling such dynamics in graphs. Embed the time sequence into each node can also be useful; for example, long-short-term-memory (LSTM) can be used on each vertex to incorporate sequential changes.

7.3 Scalability of Graph Embedding.

With the rapid growth of social networks, which contain millions and billions of nodes and edges, We expect to see graphs of a larger scale and higher diversity. How to embed large-scale graph data efficiently and accurately is still an open problem. Deep neural network models have the state-of-the-art performance. However, these methods suffer from low efficiency. They rely on modern GPU to find the optimal parameters. Better paradigms are needed for processing large-scale graphs. One possibility is to use a feed-forward machine-learning design to process the graph without BP. Another option is to use better graph coarsening or partitioning method for data pre-processing.

7.4 Interpretability of Graph Embedding.

Most state-of-the-art graph embedding methods are built upon CNNs, which are trained with backpropagation (BP) to determine the values of their model parameters. However, the training complexity is very high. Some research was performed to lower the training complexity such as quickprop [12]. However, training model parameters iteratively using BP is still time consuming and hardware demanding. In addition, the interpretability has always been an Achilles' heel of CNNs, and has presented challenges for years. Some researchers have tried to explain the interpretability of neural network models [25, 62]. The authors in [26] attempt to explain CNNs using an interpretable and feed-forward (FF) design without any BP. The work in [26] adopts a FFdata-centric approach to network parameters of the current layer based on data statistics from the output of the previous layer in a one-pass manner. It would be worthy to apply FF machine learning methods to graph embedding tasks. An interpretable design as an alternative to advanced neural network architectures can shed light on current graph embedding-related machine learning research.

Chapter 8

Conclusion

In this thesis, we have proposed a novel end-to-end graph pooling approach PagePool for graph representation learning. It extends GATs to perform graph classification instead of node classification. We have also proposed an effective way to calculate feature-aware PageRank values for node ranking in graphs. Compared to PageRank, the attentional pagerank computes node importances based not only on graph connectivity bust also on node features. Compared to other graph pooling methods, PagePool provides a simple but effective way to select the most important nodes when learning a graph-level representation suitable for the task of graph classification. In our experiments on four popular benchmarking datasets, GAT-PP achieves state-of-the-art performance. As future work, we would like to evaluate GAT-PP on more datasets and investigate which methods are most suitable for which type of datasets. We would also like to study if PagePool can be used effectively with non-GAT based graph neural networks.

Bibliography

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. Watch your step: Learning node embeddings via graph attention. *Advances in neural information processing systems*, 31, 2018.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[4] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[5] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018.

[6] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. Gram: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 787–795, 2017.

[7] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[9] Kien Do, Truyen Tran, and Svetha Venkatesh. Graph transformation policy network for chemical reaction prediction, 2018.

[10] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[11] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. *CoRR*, abs/1912.09893, 2019.

[12] Scott E Fahlman et al. *An empirical study of learning speed in back-propagation networks*. Carnegie Mellon University, Computer Science Department Pittsburgh, PA, USA, 1988.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, page 417–426, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.

[15] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[16] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1416–1424, 2018.

[17] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[18] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[19] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pages 2434–2444. PMLR, 2019.

[20] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[21] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems, 2018.

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[25] C-C Jay Kuo. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41:406–413, 2016.

[26] C-C Jay Kuo, Min Zhang, Siyang Li, Jiali Duan, and Yueru Chen. Interpretable convolutional neural networks via feedforward design. *Journal of Visual Communication and Image Representation*, 60:346–359, 2019.

[27] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[28] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[29] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[30] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, April 2016.

[31] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 125–143, Cham, 2016. Springer International Publishing.

[32] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online, July 2020. Association for Computational Linguistics.

[33] Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *Advances in Neural Information Processing Systems*, 31, 2018.

[34] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020.

[35] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. *Advances in neural information processing systems*, 31, 2018.

[36] Medhini Narasimhan, Svetlana Lazebnik, and Alexander G. Schwing. Out of the box: Reasoning with graph convolution nets for factual visual question answering, 2018.

[37] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[38] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1063–1072, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[39] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks, 2018.

[40] Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3527–3534. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[41] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[42] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control, 2018.

[43] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[44] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[45] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017.

[46] Damien Teney, Lingqiao Liu, and Anton van den Hengel. Graph-structured representations for visual question answering, 2016.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[49] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[50] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J. Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *CoRR*, abs/1909.01315, 2019.

[51] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[52] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022.

[53] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The World Wide Web Conference*, WWW '19, page 2091–2102, New York, NY, USA, 2019. Association for Computing Machinery.

[54] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[55] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[56] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*, 2018.

[57] Nuo Xu, Pinghui Wang, Long Chen, Jing Tao, and Junzhou Zhao. Mr-gnn: Multi-resolution and dual graph neural network for predicting structured entity interactions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3968–3974. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[58] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7370–7377, Jul. 2019.

[59] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

[60] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[61] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[62] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8827–8836, 2018.

[63] Yue Zhang, Qi Liu, and Linfeng Song. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[64] Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online, July 2020. Association for Computational Linguistics.

[65] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 06 2018.

Appendix A

Code

```
from torch import nn
import dgl.function as fn
from dgl.nn.functional import edge_softmax
import dgl.nn.pytorch as dglnn
from dgl import DGLGraph, topk_nodes, unbatch

class PagePooling_R(nn.Module):
    def __init__(self, k):
        super(PagePooling_R, self).__init__()
        self.k = k

    def pr_message_func(self, edges):
        return {'ap': edges.src['ap'], 'beta': edges.data['beta']}

    def pr_reduce_func(self, nodes):
        ap = th.sum(nodes.mailbox['ap'] * nodes.mailbox['beta'], dim=1)
        return {'ap': ap}

    def att_pagerank(self, g, damp):
        g.update_all(
            message_func=self.pr_message_func,
            reduce_func=self.pr_reduce_func
            )
        g.ndata['ap'] = ((1 - damp) / g.ndata['N']) + g.ndata['ap'] * damp

    def forward(self, g, feat, e, damp=0.15, max_iter=500):
        with g.local_scope():
            g.edata['e'] = e
            g.edata['beta'] = edge_softmax(g, g.edata['e'], norm_by='src')
            g.ndata['deg'] = g.out_degrees(g.nodes()).float()
            old = g.ndata['ap']
            self.att_pagerank(g, damp)
```

```
total_iter = 1
while (th.equal(old, g.ndata['ap']) != True) and
(total_iter <= max_iter):
    old = g.ndata['ap']
    self.att_pagerank(g, damp)
    total_iter += 1
g.ndata['hg'] = th.cat((feat, g.ndata['ap']), 1)
hg = topk_nodes(g, 'hg', k=self.k, sortby=-1)[0]
hg = hg.reshape(-1, 1, self.k * (feat.shape[-1]+1))
return hg
```