ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN2321-2152

www.ijmece .com

Vol 7, Issue.2 April 2019

Estimated Data Set for Partially Observed System Approximate Planning and Reinforcement Learning

Y. Prasada Reddy

Abstract

Graph representation learning (GRL) has seen a dramatic increase in interest as of late. The availability of labelled data has led to the development of three main kinds of GRL techniques. Network embedding is the first; it's concerned with the unsupervised learning of representations of relationship structures. The second one is known as graph regularised neural networks, and it incorporates a regularisation objective into neural network losses to teach semi-supervised learning via the use of graphs. To conclude, graph neural networks can learn differentiable functions across discrete topologies with any structure. Intriguingly, despite the relative popularity of these domains, there has been hardly any attempt to combine the three paradigms. Our goal here is to provide a bridge between network embedding, graph regularisation, and graph neural networks. Our comprehensive taxonomy of GRL techniques is an attempt to unite several fields of research. We focus on the GRAPHEDM framework, which brings together popular techniques for learning graph representations using semi-supervised and unsupervised approaches. These methods include GraphSage, GCN, GAT, DeepWalk, and node2vec. To show how generalizable GRAPHEDM is, we used this framework to accommodate over 30 current approaches. We believe this consolidated view serves a dual purpose: first, it sets the stage for further research in the area, and second, it clarifies the reasoning behind these methods.

Keywords: Learning on Manifolds, Relational Learning, Geometric Deep Learning, and Network Embedding

1. Introduction

2. Building models for complex structured data sets is a challenging task. There have been a great deal of successful models developed for structured data specified on discretized Euclidean domains within the last decade. Recurrent neural networks are used to represent sequential data, such as text or videos, as an example. The networks' success in machine translation and speech recognition tests is evidence of their ability to efficiently represent sequential data. Another example is convolutional neural networks (CNNs), which by parameterizing neural networks according to structural priors like shift-invariance have achieved amazing performance in pattern recognition applications like image classification and speech recognition. For certain types of data with simple relational structures, such as sequential or pattern-based data, these outstanding accomplishments have only proved useful. It is necessary to extract data from systems where the data is not always regular in order to understand the interaction between items, as complicated connection structures often arise. In addition to semi-supervised learning, social networks, computational

Department of Computer Science and Engineering
KSRM College of Engineering (A)
Kadapa

When it comes to representing complicated relational data, domains often turn to graphs, which are universal data structures comprised of nodes and edges (Gilmer et al., 2017; Stark et al., 2006; Konstas et al., 2009; Garcia and Bruna, 2018). It is challenging to build networks with robust structural priors for graph-structured data due to the fact that graph topologies aren't always constant and might vary substantially between graphs and even amongst nodes within the same graph. In particular, operations such as convolutions do not work on graph domains that are not regular. For instance, you may apply the same filter weights across the board since every pixel in
a picture has the same neighbourhood structure. But there can be no ordering of nodes (Fig. 1) since each node in a network may have its own unique neighbourhood structure. Also, geometric priors (such shift invariance) employed in Euclidean convolutions may not work in non- Euclidean domains (such domains may not even allow translations to be specified).

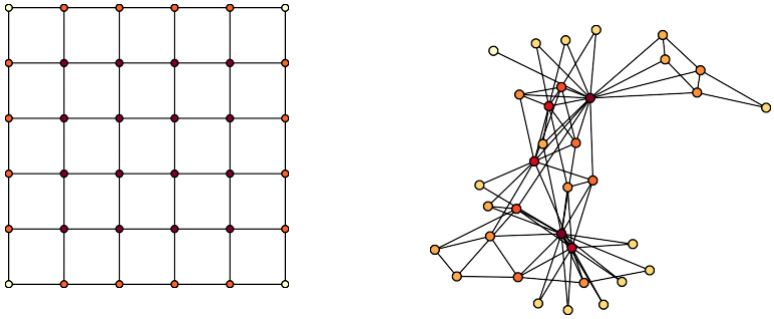
3. In reaction to these challenges, research into Geometric Deep Learning (GDL) arose; GDL aims to apply deep learning algorithms to data that is not geometrically regular. Due to the prevalence of graphs in practical contexts, many are enthusiastic about applying machine learning methods to graph-structured data. One way to describe graph-structured data is via learned embeddings, which are low-dimensional continuous vector representations. GRL approaches are one example of this. Typically, GRL learning tasks may be either supervised (or semi-supervised) or unsupervised. The first set of guidelines is predicated on the concept of acquiring low- dimensional geometrical representations that preserve the initial graph topology. The second set of predictors also learns to use low-dimensional geometrical representations for specific downstream prediction tasks like node or graph classification. Various signals described on graphs, or node properties, are usually used as inputs in the supervised setting, as opposed to the unsupervised environment, where inputs are generally graph structures. The underlying discrete graph domain can stay stable in a transductive learning context, but it can change in an inductive learning scenario (like when trying to predict molecular attributes where each molecule is a graph). Finally, it's worth noting that most supervised and unsupervised methods learn representations in geometry-based vector spaces, but that interest in learning representations that are not based on Euclidean geometry has recently increased. The goal of this learning style is to gain understanding of non-geometric embedding spaces, including spherical or hyperbolic spaces. A continuous embedding space that mimics the underlying discrete structure of the input data is the main focus of this study (e.g., hyperbolic space is a continuous version of trees; Sarkar, 2011).

Given the exponential growth of the GRL area, we believe it is essential to unify and clarify these
methods within a unified and accessible framework. With any luck, this paper will help readers have a better grasp on the many ways in which deep learning models use graph structure by offering a thorough summary of representation learning approaches for graph-structured data.

Several learning questionnaires using graph representation are accessible. The issue of shallow network embedding and auto-encoding techniques has been thoroughly reviewed in several surveys. (Cai et al., 2018; Chen et al., 2018a; Goyal and Ferrara, 2018b; Hamilton et al., 2017b; Zhang et al., 2018a) are publications that we suggest for this. Secondly, a thorough evaluation of deep learning techniques for non-Euclidean data, such manifolds or graphs, is given by Bronstein et al. (2017). Thirdly, several recent studies have explored methods that apply deep learning to graphs, namely graph neural networks (Battaglia et al., 2018; Wu et al., 2019; Zhang et al., 2018c; Zhou et al., 2018). Instead of forming connections across different domains of graph representation learning, the majority of These studies narrow their emphasis to only one. Graph Encoder Decoder Model (GRAPHEDM) is an overarching paradigm that categorises existing research into four key areas: (i) techniques for shallow embedding, (ii) methods for auto- encoding, (iii) methods for graph regularisation, and (iv) approaches for graph neural networks (GNNs). Hamilton et al. (2017b) presented an encoder-decoder model, which this framework builds upon.Additionally, we provide a Graph Convolution Framework (GCF) to describe GNNs

We may compare and analyse several GNNs with different designs, as stated by
Veli˘ckovi'c et al. (2018). Some of these GNNs employ self-attention as a neighbourhood aggregation function, while others work in the Graph Fourier1 domain. In order to help readers better grasp the various graph-

based learning techniques, this thorough summary of current research aims to highlight their commonalities and distinctions, as well
as its potential expansions and limitations. But our poll is different from the others in three key respects:
We introduce a general framework, GRAPHEDM, to describe a broad range of super- vised and unsupervised methods that operate on graph-structured data, namely shal- low embedding methods, graph regularization methods, graph auto- encoding methods and graph neural networks.
Our survey is the first attempt to unify and view these different lines of work fromthe same perspective, and we provide a general taxonomy (Fig. 3) to understand differences and similarities between these methods. In particular, this taxonomy en-



(a) Grid (Euclidean). (b) Arbitrary graph (Non-Euclidean).

Figure 1: An illustration of Euclidean vs. non-Euclidean graphs.

stands for over 30 distinct GRL algorithms. A detailed taxonomy may help us comprehend the distinctions between different tactics.

Link prediction and node classification are two of the important graph applications that our open-source GRL library offers. The library also features state-of-the-art GRL algorithms. At https://github.com/google/gcnn- survey-paper, you may discover our implementation.Anyone may come and see it.Survey administration In Section 2, we go over some basic graph principles and provide a clear explanation of the problem context for GRL. In Section 2.2.1, the role of node features in GRL and their connection to supervised GRL are discussed. In Section 2.2.2, inductive and transductive learning are defined. In Section 2.2.3.1, positional and structural embeddings are differentiated. Finally, in Section 2.2.4, supervised and unsupervised embeddings are defined. We go on to explain what these key ideas in GRL are and how they vary from one another. The third section introduces GRAPHEDM, a general framework that may be used in both inductive and transductive learning settings to specify supervised and unsupervised GRL methods, regardless of whether they include or do not include any nodes. Based on GRAPHEDM, which includes over thirty modern GRL models, we provide a thorough taxonomy of GRL methods (Fig. 3). In Section 5, we describe supervised approaches, and in Section 4, we describe unsupervised methods, using this taxonomy. Graph applications are reviewed in Section 6, which is the last section.

3. Preliminaries

4. Graph representation learning approaches attempt to address the generalized network embeddingissue; for an overview, see Table 1. Here, we offer the notation used throughout the article.

ISSN2321-2152

www.ijmece .com

Vol 7, Issue.2 April 2019

6.1 Definitions

	Notation	Meaning
Abbreviations	GRL GRAPHEDM GNN GCF	Graph Representation Learning Graph Encoder Decoder Model Graph Neural Network Graph Convolution Framework
Graph notation	$G = (V, E)$ $v_i \in V$ $d_G(\cdot, \cdot)$ $\deg(\cdot)$ $D \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$ $W \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$ $\widetilde{W} \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$ $A \in \{0,1\}^{\lvert V\rvert \times \lvert V\rvert}$ $L \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$ $\tilde{L} \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$ $L^{rw} \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$	Graph with vertices (nodes) V and edges E Graph vertex Graphdistance (length of shortest path) Nodedegree Diagonal degree matrix Graph weighted adjacency matrix Symmetric normalized adjacency matrix $(\widetilde{W} = D^{-1/2}WD^{-1/2})$ Graph unweighted weighted adjacency matrix Graph unnormalized Laplacian matrix $(L = D - W)$ Graph normalized Laplacian matrix $(\tilde{L} = I - D^{-1/2}WD^{-1/2})$ Random walk normalized Laplacian $(L^{rw} = I - D^{-1}W)$

ISSN2321-2152

www.ijmece .com

Vol 7, Issue.2 April 2019

Table 1: Summary of the notation used in the paper.

	d_0	Input feature dimension
	$X \in \mathbb{R}^{\lvert V\rvert \times d_0}$	Node feature matrix
	d	Final embedding dimension
	$Z \in \mathbb{R}^{\lvert V\rvert \times d}$	Node embedding matrix
	d_l	Intermediate hidden embedding dimension at layer l
	$H^l \in \mathbb{R}^{\lvert V\rvert \times d_\epsilon}$	Hidden representation at layer l
	$\mathcal{Y}$	Label space
	$y^S \in \mathbb{R}^{\lvert V\rvert \times \lvert\mathcal{Y}\rvert}$	Graph($S = G$) or node ($S = N$) ground truth labels Predictedlabels
	$\hat{y}^S \in \mathbb{R}^{\lvert V\rvert \times \lvert\mathcal{Y}\rvert}$	Target similarity or dissimilarity matrix in graph regularization
	$s(W) \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$	Predicted similarity or dissimilarity matrix
GRAPHEDM notation	$\lvert\hat{W}\rvert \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$	Encoder network with parameters Θ^EGraph
	$\text{ENC}(\cdot\,; \Theta^E)$	decoder network with parameters Θ^D Label
	$\text{DEC}(\cdot\,; \Theta^D)$	decoder network with parameters Θ^S
	$\text{DEC}(\cdot\,; \Theta^S)$	Supervised loss
	$L^S_{\text{SUP}}(y^S, \hat{y}^S; \Theta)$	Graph regularization loss
	$L_{G,\text{REG}}(W, \hat{W}\,; \Theta)$	Parameters' regularization loss
	$L_{\text{REG}}(\Theta)$	Matrix distance used for to compute the graph regularization loss
	$d_1(\cdot, \cdot)$	Embedding distance for distance-based decoders
	$d_2(\cdot, \cdot)$	
	$\lVert\cdot\rVert_p$	$p-$norm
	$\lVert\cdot\rVert_F$	Frobenuis norm

7. A Taxonomy of Graph Embedding Models

We first describe our proposed framework, GRAPHEDM, a general framework for GRL (Sec- tion 3.1). In particular, GRAPHEDM is general enough that it can be used to

ISSN2321-2152

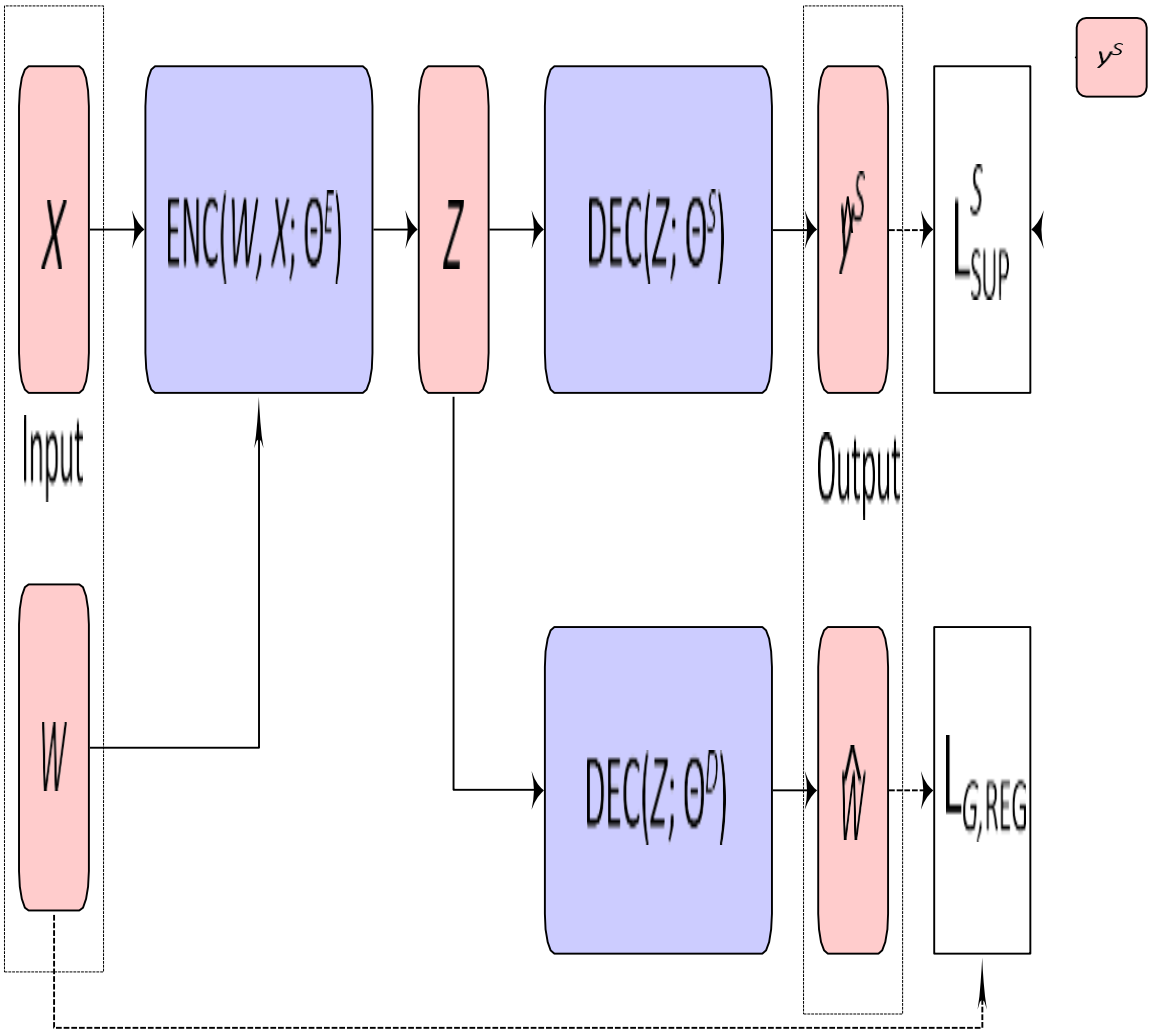
www.ijmece .com

Vol 7, Issue.2 April 2019

succinctly de- scribe over thirty GRL methods (both unsupervised and supervised). We use GRAPHEDMto introduce a comprehensive taxonomy in Section 3.2 and Section 3.3, which summarizes exiting works with shared notations and simple block diagrams, making it easier to under- stand similarities and differences between GRL methods.

7.1 The GraphEDM framework

The GRAPHEDM framework builds on top of the work of Hamilton et al. (2017b), which describes unsupervised network embedding methods from an encoder-decoder



perspective.

Figure 2: Illustration of the GRAPHEDM framework. Based on the supervision available, methods will use some or all of the branches. In particular, unsupervised methods do not leverage label decoding for training and only optimize the similarity or dissimilarity decoder (lower branch). On the other hand, semi-supervised and supervised methods leverage the additional supervision to learn models' parameters (upper branch).

Cruz et al. (2019) also recently proposed a modular encoder-based framework to describe and compare unsupervised graph embedding methods. Different from these unsupervised frameworks, we provide a more general framework which additionally encapsulates super- vised graph embedding methods, including ones utilizing the graph as a regularizer (e.g. Zhuand Ghahramani (2002)), and graph neural networks such as ones based on message passing (Gilmer et al., 2017; Scarselli et al., 2009) or graph convolutions (Bruna et al., 2014; Kipf andWelling, 2016a).

Input The GRAPHEDM framework takes as input an undirected weighted graph $G = (V, E)$, with adjacency matrix $W \in \mathrm{R}^{|V|\times|V|}$, and optional node features $X \in \mathrm{R}^{|V|\times d_0}$. In (semi-)supervised settings, we assume that we are given training target labels for nodes (denoted N), edges (denoted E), and/or for the entire graph (denoted G). We denote the supervision signal as $S \in \{N, E, G\}$, as presented below.

Model The GRAPHEDM framework can be decomposed as follows:

Graph encoder network $\mathrm{ENC}_{\Theta^E} : \mathrm{R}^{|V|\times|V|} \times \mathrm{R}^{|V|\times d_0} \rightarrow \mathrm{R}^{|V|\times d}$, parameterized by Θ^E, which combines the graph structure with node features (or not) to produce node embedding matrix $Z \in \mathrm{R}^{|V|\times d}$ as:
$Z = \mathrm{ENC}(W, X; \Theta^E)$.
As we shall see next, this node embedding matrix might capture different graph prop-erties depending on the supervision used for training.

Graph decoder network $\mathrm{DEC}_{\Theta^D} : \mathrm{R}^{|V|\times d} \rightarrow \mathrm{R}^{|V|\times|V|}$, parameterized by Θ^D, which uses the node embeddings Z to compute similarity or dissimilarity scores for all node pairs, producing a matrix $\hat{W} \in \mathrm{R}^{|V|\times|V|}$ as:
$\hat{W} = \mathrm{DEC}(Z; \Theta^D)$.

Classification network $\mathrm{DEC}_{\Theta^S} : \mathrm{R}^{|V|\times d} \rightarrow \mathrm{R}^{|V|\times|\mathcal{Y}|}$, where $\mathcal{Y}$ is the label space. This network is used in (semi-)supervised settings and parameterized by Θ^S. The output is a distribution over the labels $\hat{y}^S$, using node embeddings, as:

Our GRAPHEDM framework is general (see Fig. 2 for an illustration). Specific choices of the aforementioned (encoder and decoder) networks allows GRAPHEDM to realize specific graphembedding methods. Before presenting the taxonomy and showing realizations of various methods using our framework, we briefly discuss an application perspective.

OutputThe GRAPHEDM model can return a reconstructed graph similarity or dissim- ilarity matrix $\hat{W}$ (often used to train *unsupervised* embedding algorithms), as well as a output labels $\hat{y}^S$ for *supervised* applications. The label output space $\mathcal{Y}$ varies depending on the supervised application.

Node-level supervision, with $y^N \in \mathcal{Y}^{|V|}$, where $\mathcal{Y}$ represents the node label space. If $\mathcal{Y}$ is categorical, then this is also known as (semi-)supervised node classification (Section 6.2.1), in which case the label decoder network produces labels for each node in the graph. If the embedding dimensions d is such that $d = |\mathcal{Y}|$, then the label decoder network can be just a simple softmax activation across the rows of Z, producing a distribution over labels for each node. Additionally, the graph decoder network might also be used in supervised node-classification tasks, as it can be used to regu- larize embeddings (e.g. neighbor nodes should have nearby embeddings, regardless ofnode labels).

Edge-level supervision, with $y^E \in \mathcal{Y}^{|V|\times|V|}$, where $\mathcal{Y}$ represents the edge label space. For example, $\mathcal{Y}$ can be multinomial in knowledge graphs (for describing the then rather than naming the output of the decoder as $\hat{}$, we instead follow the nomenclature and position link prediction as an *unsupervised* task (Section 4). Then inlieu of y^E we utilize $\hat{W}$, the output of the graph decoder network (which is learned to reconstruct a target similarity or dissimilarity matrix) to ran potential edges. the graph classification task (Section 6.2.2), the label decoder network converts node embeddings into a single graph labels, using *graph pooling* via the graph edges capturedby W. More concretely, the graph pooling operation is similar to pooling in standard CNNs, where the goal is to downsample local feature representations to capture higher-level information. However, unlike images, graphs don't have a regular grid structure and it is hard to define a pooling pattern which could be applied to every node in the graph. A possible

way of doing so is via graph coarsening, which groups similar nodes into clusters to produce smaller graphs (Defferrard et al., 2016). There exist other pooling methods on graphs such as DiffPool (Ying et al., 2018b) or SortPooling (Zhang et al., 2018b) which creates an ordering of nodes based on their structural rolesin the graph. Details about graph pooling operators is outside the scope of this workand we refer the reader to recent surveys (Wu et al., 2019) for a more in-depthtreatment.

Supervised loss term, $\mathcal{L}^{S}$, which compares the predicted labels $y\hat{}^{S}$ to the ground truth labels y^{S}. This term
SUP
depends on the task the model is being trained for. For instance, in semi-supervised node classification tasks (S =
N), the graph vertices are

split into labelled and unlabelled nodes ($V = V_L \cup V_U$), and the supervised loss is computed for each labelled node in the graph:$\mathcal{L}_N$ $(y^N, y\hat{}^N; \Theta)$ =

where $\boldsymbol{l}(\cdot)$ is the loss function used for classification (e.g. cross-entropy). Similarly for graph classification tasks (S = G), the supervised loss is computed at the graph-level and can be summed across multiple training graphs: ^{G} ^{G}, $y\hat{}^{G}; \Theta) = l(y^{G}, y\hat{}^{G}; \Theta)$.

Graph regularization loss term, $\mathcal{L}_{G,\text{REG}}$, which leverages the graph structure to impose regularization constraints on the model parameters. This loss term acts as a smoothing term and measures the distance between the decoded similarity or dissim- ilarity matrix $\hat{W}$, and a target similarity or dissimilarity matrix $s(W)$, which might capture higher-order proximities than the adjacency matrix itself:

$$\mathcal{L}_{G,\text{REG}}(W, \hat{W}; \Theta) = d_1(s(W), \hat{W}) , \quad (1)$$

where $d_1(\cdot, \cdot)$ is a distance or dissimilarity function. Examples for such regularization are constraining neighboring nodes to share similar embeddings, in terms of their dis- tance in L2 norm. We will cover more examples of regularization functions in Section 4and Section 5.

Weight regularization loss term, $\mathcal{L}_{\text{REG}}$, e.g. for representing prior, on trainable model parameters for reducing overfitting. The most common regularization is L2 regularization (assumes a standard Gaussian prior):
Σ
$\hat{}$

$$\mathcal{L} = \alpha\mathcal{L}_{S\text{SUP}}(y , y\hat{} ; \Theta) + \beta\mathcal{L}_{G,\text{REG}}(W, W ; \Theta) + \gamma\mathcal{L}_{\text{REG}}(\Theta), \quad (2)$$

where α, β and γ are hyper-parameters, that can be tuned or set to zero. Note that
graphembedding methods can be trained in a *supervised* (α /= 0) or *unsupervised* (α = 0) fashion.

Supervised graph embedding approaches leverage an additional source of information to learn embeddings such as node or graph labels. On the other hand, unsupervised network embedding approaches rely on the graph structure only to learn node embeddings.
A common approach to solve supervised embedding problems is to first learn embeddingswith an unsupervised method (Section 4) and then train a supervised model on the learned embeddings. However, as pointed by Weston et al. (2008) and others, using a two-step learning algorithm might lead to sub-optimal performances for the supervised task, and in general, supervised methods (Section 5) outperform two-step approaches.

Taxonomy of encoders

Having introduced all the building blocks of the GRAPHEDM framework, we now introduce our graph embedding taxonomy. While most methods we describe next fall
We divide graph embedding models into four main categories:
Shallow embedding methods, where the encoder function is a simple embedding lookup. That is, the parameters of the model Θ^{E} are directly used as node embed- dings: Note that shallow embedding methods rely on an embedding lookup and are therefore *transductive*, i.e. they generally cannot be directly applied in *inductive* settings where the graph structure is not fixed.
Graph regularization methods, where the encoder network ignores the graph structure and only uses node features as input:
$Z = \text{ENC}(X; \Theta^{E})$.
As its name suggests, graph regularization methods leverage the graph structure
through the graph regularization loss term in Eq. (2) (β $\neq$ 0) to regularize node embeddings.
Graph auto-encoding methods, where the encoder is a function of the graph structure only:
$Z = \text{ENC}(W; \Theta^{E})$.
Neighborhood aggregation methods, including graph convolutional methods, where both the node features and the graph structure are used in the encoder network. Neighborhood aggregation methods use the graph structure to propagate informationacross nodes and learn embeddings that encode structural properties about the graph:

Historical Context

There is a general two-step process that most machine learning models adhere to. Initially, they forego the

need of human feature building in favor of automatically extracting significant patterns from data. According to Bengio et al. (2013), this is the part where representation learning takes place. A second step involves putting these representations to use in supervised (like classification) or unsupervised (like clustering, visualization, and nearest-neighbor search) applications further down the line. This task is referred to as downstream processing.3 To facilitate the downstream process, a good data representation should be both expressive and concise, preserving the original data's significant qualities. Overfitting and other problems induced by the curse of dimensionality may be mitigated, for example, by using low-dimensional representations of high-dimensional datasets. When it comes to GRL, a graph encoder is used for representation learning, while a graph or label decoder is employed for jobs further down the line, such as node classification and link prediction. Graph encoder-decoder networks have traditionally been used for manifold learning. It is usual to presume that input data, even if it exists on a high- dimensional Euclidean space, is inherently contained on a low-dimensional manifold. The classic manifold hypothesis describes this. This inherently low-dimensional manifold is what manifold learning methods aim to retrieve. A discrete approximation of the manifold is often constructed initially, in the form of a graph with edges connecting adjacent points in the ambient Euclidean space.Graph distances are a reasonable surrogate for local and global manifold distances because manifoldsare locally Euclidean. Secondly, while keeping graph distances as accurate as feasible, "flatten" this representation of the graph by learning a non-linear mapping from graph nodes to points in low- dimensional Euclidean space. Typically, these representations are more manageable compared to theinitial high-dimensional ones, and they may subsequently be used in subsequent **ISSN2321-2152**
When looking for solutions to the manifold learning issue, non-linear4 dimensionality reduction strategies were all the rage in the early 2000s. For example, spectral approaches are used by Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2002) to calculate embeddings, and IsoMap (Tenenbaum et al., 2000) to maintain global network geodesics by a mix of the Floyd-Warshall algorithm and the conventional Multi-dimensional scaling algorithm. In Section 4.1.1, we outline a few of these techniques that use shallow encoders. Despite their significant influence on machine learning, manifold dimensionality reduction approaches are not scalable to big datasets. Consider the time complexity of IsoMAP: it exceeds quadratic time due to the need to compute all pairs of shortest pathways. Since the mappings from node to embeddings are non-parametric, they cannot generate embeddings for additional datapoints, which is a potentially more significant drawback.The issue of graph embedding has seen several proposals for non- shallow network topologies in recent years. Our GRAPHEDM framework may be used to define graph neural networks and graph regularization networks. When compared to traditional approaches, GRL models often provide more expressive, scalable, and generalizable embeddings due to their use of deep neural networks' expressiveness.
the next sections, we review recent methods for supervised and unsupervised graphembedding techniques using GRAPHEDM and summarize the proposed taxonomy in Fig. 3.

Unsupervised Graph Embedding
Using the taxonomy outlined earlier, we will now provide a summary of current methods forunsupervised graph embedding. Without using task-specific labels for the network or its nodes, theseapproaches map the graph into a continuous vector space, including its edges and/or nodes. By learning to rebuild matrices that measure the similarity or dissimilarity between nodes, such as the adjacency matrix, some of these approaches aim to learn embeddings that maintain the network structure. There are methods that use a contrastive objective. For example, one could compare nearbynode-pairs to faraway ones: nodes that are co-visited in short random walks should have a higher similarity score than distant ones. Another would compare real graphs to fake ones: the mutual information between a graph and all of its nodes should be higher in real graphs than in fake ones.

Shallow embedding methods
The encoder function in shallow embedding techniques is a basic embedding lookup; these methods are transductive graph embedding methods. The shallow encoder function is simply: for every node vi in V, there is a corresponding low-dimensional learnable embedding vector Zi in Rd.The data structure in the embedding space matches the underlying graph structure, thanks to learnt node embeddings. Generally speaking, it's not dissimilar to principal component

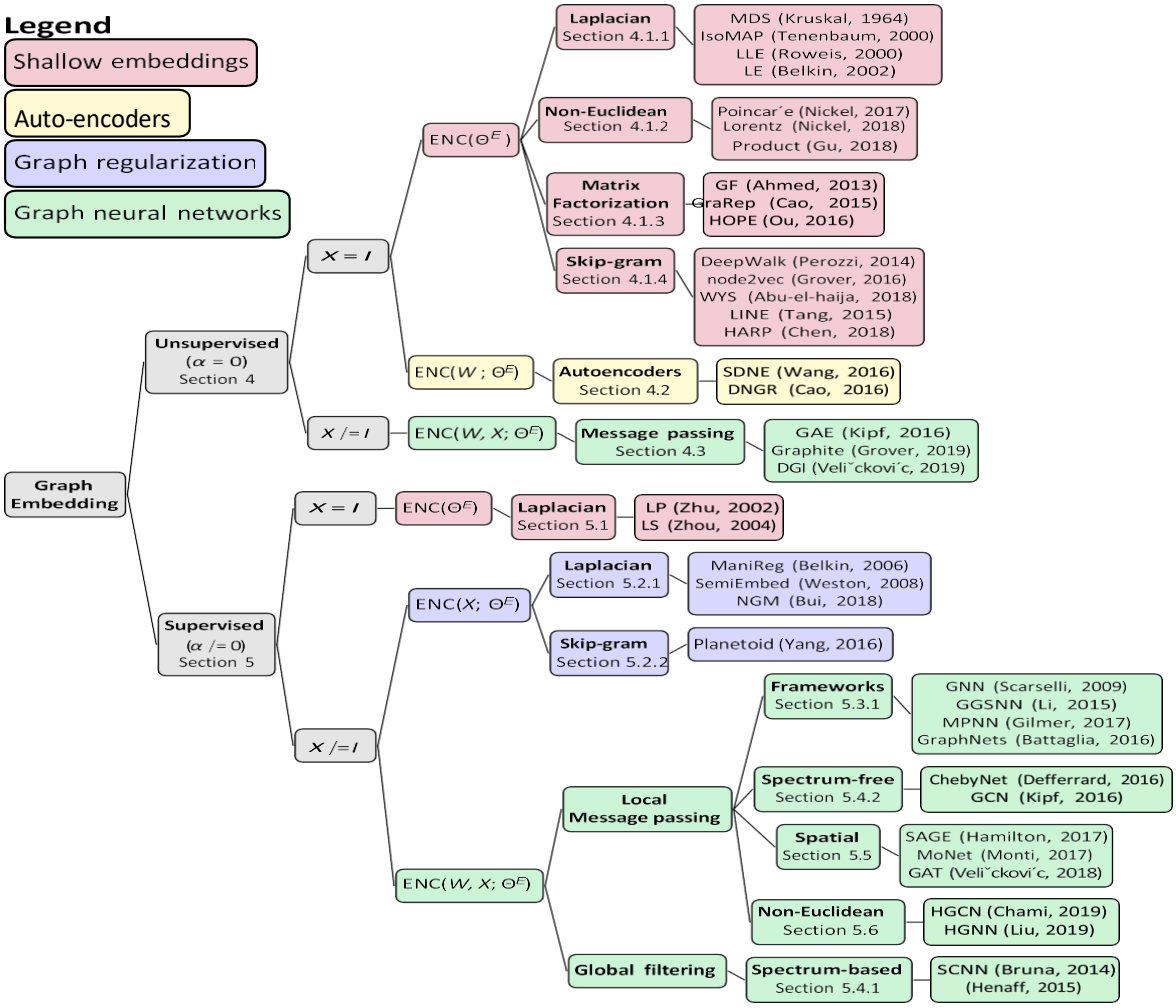
ISSN2321-2152

www.ijmece .com

Vol 7, Issue.2 April 2019

analysis (PCA) and other dimensionality reduction techniques; however, the input data may not be linear. Specifically, graph embedding issues may be addressed usingtechniques for non-linear dimensionality reduction, which often begin with constructing a discrete graph from the data in order to approximate the manifold. We take a look at the distance-based and outer product-based approachesto shallow graph embedding.

Distance-based methods By using a preset distance function, these approaches maximize embeddings in a way that keeps points that are close together in the graph (as shown by their graph distances, for example) as near togetherin the embedding space as feasible. In a formal sense, the decoder network may provide either non-Euclidean (Section 4.1.2) or Euclidean (Section 4.1.1) embeddings by computing pairwise distance for a certain distance function d2:



$$\hat{W} = \mathrm{DEC}(Z; \Theta^D)$$

with $\hat{W}_{ij} = d_2(Z_i, Z_j)$

Figure 3: Taxonomy of graph representation learning methods. Based on what informationis used in the encoder network, we categorize graph embedding approaches into four cat-

egories: shallow embeddings, graph auto-encoders, graph-based regularization and graph neural networks. Note that message passing methods can also be viewed as spatial convolution, since messages are computed over local neighborhood in the graph domain. Reciprocally, spatial convolutions can also be described using message passing frameworks.

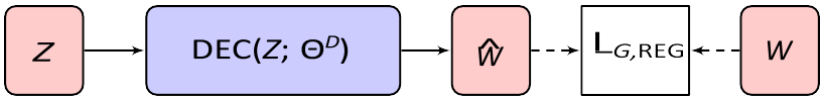


Figure 4: Shallow embedding methods. The encoder is a simple embedding look-up andthe graph structure is only used in the loss function.

Outer product-based methods These methods on the other hand rely on pairwisedot-products to compute node similarities and the decoder network can be written as:

Embeddings are then learned by minimizing the graph regularization loss: $L_{G,REG}(W, \hat{W}; \Theta) = d_1(s(W), \hat{W})$. Note that for distance-based methods, the function $s(\cdot)$ measures dissimilar- ity or distances between nodes (higher values mean less similar pairs of nodes), while in outer-product methods, it measures some notion of similarity in the graph (higher values mean more similar pairs).

4.1.1

ISSN2321-2152

www.ijmece .com

Vol 7, Issue.2 April 2019

DISTANCE-BASED: EUCLIDEAN METHODS

In order to optimise Euclidean embeddings, most distance-based methods aim to minimise the Euclidean distance between similar nodes. Nonlinear methods such as Laplacian eigenmaps, IsoMAP, and local linear embedding are available, in addition to linear embedding techniques that generate linear projection subspaces in low dimensions, such as principal component analysis (PCA) and maximum degree of separation (MDS). Although these methods were first introduced for dimensionality reduction or visualisation, they may be easily used for graph embedding as well.

Multi-Dimensional Scaling

The majority of distance-based approaches to optimising Euclidean embeddings seek to reduce the Euclidean distance between nodes that are comparable. Laplacian eigenmaps, IsoMAP, and local linear embedding are nonlinear approaches; principal component analysis (PCA) and maximum degree of separation (MDS) are linear embedding techniques that produce low-dimensional linear projection subspaces. These techniques may be readily used to graph embedding as well, despite their initial introduction for dimensionality reduction or visualisation.

$_{ij} = d_2(Z_i, Z_j) = ||Z_i - Z_j||_2.$

$_{ij}\ s(W)^2$

To rephrase, mMDS finds an embedding configuration that maintains distances in the low-dimensional embedding space using the stress cost function, which is a residual sum of squares. The mMDS and PCA
dimensionality reduction methods are identical when a higher-dimensional representation's Euclidean distances are used to compute the dissimilarities. Finally, there are variants of this method. A low-rank decomposition of the gramme matrix allows for closed-form execution of classical MDS (cMDS), and non-

15.1.1 NODE CLASSIFICATION

16. 16. Node classification, which seeks to create node representations that can accurately anticipate their labels, is a crucial supervised graph application. One possible use for node labels in citation networks is to denote scientific topics; in social networks, they may denote gender or other attributes. Since labelling massive graphs requires a significant time and financial investment, semi-supervised node classification is a common use case. In semi-supervised settings, the goal is to forecast unlabeled node properties using node links, with just a small fraction of nodes being labelled. It is deemed transductive since there is only one partially labelled fixed graph in this setting. Alternatively, you may use inductive node classification, which involves figuring out how to classify nodes in various networks. A node's performance on categorised nodes tasks may be substantially enhanced if its features accurately describe the objective label. Modern methods like GCN (Kipf and Welling, 2016a) and GraphSAGE (Hamilton et al., 2017a) have achieved state- of-the-art performance on multiple node classification benchmarks by combining

structural data with semantic information obtained from features. On the other hand, methods like random walks on graphs fail miserably at these tasks because they don't employ feature information.

21.1.1 GRAPH CLASSIFICATION

17. Graph classification is an example of a supervised application; it takes an input graph and aims to predict labels at the graph level. Graph classification issues are essentially inductive since new graphs are introduced throughout testing. As for other popular options, there are biochemical pursuits and online social networks. Molecular graphs have widespread use in biology. Nodes in these graphs may be feature vectors that encode the number of atoms in a 1-hot fashion, and edges between nodes can represent bonds, with the feature vector indicating the kind of bond. (Debnath et al., 1991) MUTANG is a task-dependent graph-level label that shows if a drug is mutagenic to bacteria. Online social networks often use the node metaphor, with connections and interactions depicted by the edge metaphor. Graph categorization tasks on Reddit, for instance, include a large number of graphs (Yanardag and Vishwanathan, 2015). When one person responds to another's comment, for example, an edge will connect the two nodes in the discussion thread graph. Finding the community (sub-reddit) where a discussion took place is the goal, given a comment graph.

Graph classification tasks need a new sort of pooling to aggregate data at the node and graph levels, in contrast to edge prediction and node classification, which both use pooling at the edge level. Extending the idea of pooling to any kind of graph is a difficult and continuing area of research, as mentioned before. We want the pooling mechanism to be unaffected by node order. Some methods use simple pooling, such as adding up all the latent vectors at the network node level or taking the mean of them (Xu et al., 2018). Approaches that use differentiable pooling include Ying et al., 2018b; Cangea et al., 2018; Gao and Ji, 2019; and Lee et al., 2019. To name a few. Supervisorial techniques for learning graph-level representations are provided by Tsitsulin et al. (2019), Al-Rfou et al. (2019), and Tsitsulin et al. (2020a), but other unsupervised methods are also available. analysed as graph kernels (GKs) by Viswanathan et al. (2010) and Kriege et al. (2020). Although GKs are not our primary concern, we do touch on their links to GRAPHEDM here. Graph-level tasks,

such graph categorization, are suitable for GKs. In order to convert any two graphs into a scalar, GK may automatically apply a similarity function. Counting the number of walks (or pathways) that two graphs have in common is one way that traditional GKs calculate graph similarity. For example, each walk may be stored as a seriesof node labels. Common practice dictates using node degrees as labels in the absence of explicit labels. The capacityof GKs to identify (sub-)graph isomorphism is a common metric for analysis. When ordering of nodes is ignored, two (sub-)graphs are considered isomorphic if they are identical. According to the 1-dimensional Weisfeiler-Leman(1-WL) heuristic, two sub-graphs are considered isomorphic since sub-graph isomorphism is NP-hard. In each graph, histograms are used to tally the statistics of the nodes (e.g., how many nodes with the label "A" have an edge to nodes with the label "B"). If two graphs' histograms, obtained from the same 1-hop neighborhood, are equal, thenthe graphs are considered isomorphic according to the 1-WL heuristic. An example of a GNN that has been shownto achieve the 1-WL heuristic is the Graph Isomorphism Network (GIN; Xu et al., 2018). This means that GIN canonly map two graphs to the same latent vector if they are considered isomorphic according to the 1-WL heuristic. In some newer studies, GKs and GNNs are used together. Using the similarity of the "tangent space" of the goal with respect to the Gaussian-initialized GNN parameters, Du et al. (2019) models the similarity of two graphs, andChen et al. (2020) extracts walk patterns. There isn't any GNN training in either (Du et al., 2019; Chen et al., 2020).Instead, kernel support vector machines and other kernelized algorithms are

used to the pairwise Gram matrix during training. Therefore, our GCF and GRAPHEDM frameworks are not well-suited to include these methodologies. However, there are other approaches that don't rely on indirectly computing graph-to-graph similarity scalar scoresbut instead directly map graphs to high-dimensional latent spaces. One example is Morris et al.'s (2019) k-GNN network, which is deliberately coded as a GNN but can actually implement the k-WL heuristic (which is identical to 1-WL but where histograms are produced up-to k-hop neighbors). Therefore, our GCF and GRAPHEDM frameworks can define the k-GNN model classes.

Conclusion and Open Research Directions

In this review, we laid forth a common procedure for evaluating ML models that have been trained on graph- structured data. Our enhanced GRAPHEDM system, which was previously used for unsupervised network embedding, now incorporates deep graph embedding methods, graph auto-encoders, graph regularisation approaches, and graph neural networks. Furthermore, we introduced a graph convolution framework (GCF) that allows us to characterise and compare convolution-based graph neural networks, including spectral and spatial graph convolutions. Our comprehensive taxonomy of GRL approaches, presented using this paradigm, included over 30 supervised and unsupervised strategies for graph embedding.If this survey is successful in its aims, it will hopefully motivate researchers to dig further into GRL, which should eventually provide answers to the issues these models are facing. As far as taxonomy is concerned, there are 19. Here, we laid the groundwork for a uniform method of comparing ML models that have been trained on graph-structured data. Now including deep graph embedding techniques, graph auto-encoders, graph regularisation approaches, and graph neural networks, our improved GRAPHEDM system may be utilised for supervised network embedding as well as unsupervised. We also presented a graph convolution framework (GCF) for comparing and characterising graph neural networks that use convolution, as well as spectral and spatial graph convolutions. More than 30 supervised and unsupervised solutions for graph embedding were included in our exhaustive taxonomy of GRL approaches, which was presented using this paradigm.With any luck, the results of this poll will encourage further investigation into method for every specific situation. Additionally, it is worth noting that scholars who have only The taxonomy may help researchers identify appropriate methodologies for data analysis, organise their questions, find relevant literature, and set trustworthy baselines for comparison. For all the success that GRL methods have had with node categorization and connection prediction, there remain a number of problems that need fixing. The next section covers the research opportunities and challenges associated with graph embedding models.

Evaluation and benchmarks

The methods discussed in this study are often evaluated using industry-standard standards for node categorization or link prediction. It is common practice to compare graph embedding methods to citation networks in order to demonstrate the argument. A concern with these small citation standards, as shown in recent study (Shchur et al., 2018), is that the results could vary substantially based on the datasets' splits or training procedures (such early stopping). In order to enhance GRL techniques, it is essential to use robust and consistent evaluation procedures and to broaden the area of assessment beyond standards for link prediction and small node classification. Examples of recent advancement in this method include graph embedding libraries (Fey and Lenssen, 2019; Wang et al., 2019; Goyal and Ferrara, 2018a), new graph benchmarks with leaderboards (Hu et al., 2020; Dwivedi et al., 2020), and other similar works. Similarly, Sinha et al. (2020) proposed a set of first-order logic tasks to evaluate GNNs' reasoning abilities.

Fairness in Graph Learning A new field known as Fairness in Machine Learning is emerging to address the issue of models associating'sensitive' attributes with the model's anticipated outcome

(Mehrabi et al., 2019). For problems with graph learning, these factors may be especially important because of the correlation between the output and the graph's structure (the edges) and the nodes' feature vectors. According to Bose Hamilton (2019), the most common way to include fairness constraints into models is via adversarial learning. To make sure the model doesn't have any bias when it comes to the sensitive feature(s), you may use this strategy to GRL. However, just how much prejudice can be eradicated by adversarial methods is an uncertain matter. Even with a combination of debiasing techniques, the task at hand could be challenging to complete (Gonen and Goldberg, 2019). Recent research in the area has concentrated on proving assurances for debiasing GRL (Palowitch and Perozzi, 2019).

Application to large and realistic graphs When dealing with datasets that include tens of thousands to hundreds of thousands of nodes, graph learning approaches are usually reserved. But there are real-world graphs that are far larger, with billions of nodes. Scalable solutions for large graphs need a Distributed Systems setup with several processors, such as MapReduce (Dean and Ghemawat, 2008; Lerner et al., 2019; Ying et al., 2018a). Is it possible for a researcher to use a home computer to implement a learning strategy on a massive graph that exceeds the capacity of random access memory (RAM) but still fits on a single hard drive (e.g., with a one terabyte size)? Compare it to other approaches that have been proposed to solve a computer vision problem utilising a big dataset (Deng et al., 2009; Kuznetsova et al., 2020). You may use RAM for whatever model you like.

learned on desktop PCs, no matter how big the dataset is. Some graph embedding models may have a hard time fixing this problem, especially if their parameters increase in size along with the graph's nodes.At times, it could be tough for businesspeople to choose the proper graph to use as input. In their description of the Google system Grale, Halcrow et al. (2020) demonstrate how it learns the correct graph from several attributes. When learning graphs from large datasets, Grale use similarity search techniques, including locality sensitive hashing. In order to facilitate end-to-end learning, Rozemberczki et al. (2021) upgraded the Grale model by include an attention

We expect new mathematical and practical challenges to arise from learning algorithms for large networks that can be executed on a single machine. We hope that academics will give this field the attention it deserves so that neurology researchers and other non-specialist practitioners may utilise these learning techniques to assess the sub-graph of the human brain, which is made up of neurons and synapses.

Molecule generation Molecular biologists may be able to save time and money by learning on graphs, which might revolutionise their work. Several approaches have been suggested by researchers to forecast the quantum characteristics of molecules (Gilmer et al., 2017; Duvenaud et al., 2015) and to create molecules with certain desired features (Liu et al., 2018; De Cao and Kipf, 2018; Li et al., 2018; Simonovsky and Komodakis, 2018; You et al., 2018). In (Elton et al., 2019), a survey of current approaches is provided. Jin et al. (2018), Ragoza et al. (2017), and Feng et al. (2018) are only a few examples of the approaches that are involved in drug design, while others are focused with producing materials with certain qualities, such as conductivity and malleability.

Combinatorial optimization

Computationally difficult problems arise in many areas, including routing science, cryptography, decision- making, and planning, among many others. The methods used to determine the optimal solution to computationally challenging issues are not very scalable. For a rundown of the methods that have lately garnered interest in handling combinatorial optimisation problems using machine learning techniques, such

or NP-hard problems, graph embeddings have recently attracted attention (Khalil et al., 2017; Nowak et al., 2017; Selsam et al., 2018; Prates et al., 2019). In reality, many problems may be represented in terms of graphs; graphs provide a suitable representation for many challenging difficulties, including SAT and vertex cover.

Computingly challenging problems may be solved using data-driven approaches, such as finding out whether a particular instance (e.g., node) is a portion of the optimal solution out of several instances of the problem. Seek for tasks in prior research on optimising graph partitions that aim to achieve a goal (e.g., the smallest conductance cut) (Bianchi et al., 2020; Tsitsulin et al., 2020b). Because GNNs are better at depicting graphs than normal neural networks (e.g., permutation invariance), these approaches all begin with GNNs. This is because GNNs have relational inductive biases. Here are several solutions that still beat Whilst these methods rely on data, GNNs have shown potential when used to more complex problems (Nowak et al., 2017; Prates et al., 2019). Lamb et al. (2020) offers a thorough synopsis of GNN-based combinatorial optimisation methods in their most recent research on neural symbolic learning.

Non-Euclidean embeddings The underlying space geometry is an important part of graph embeddings, as we saw in Sections 4.1.2 and 5.6. All graphs are discrete complex, non-Euclidean structures with high dimensions; however, there is currently no simple method for encoding such data into embeddings with low dimensions that maintainthe graph topology (Bronstein et al., 2017). Hyperbolic and mixed-product space embeddings are two examples of non-Euclidean embeddings that have recently attracted attention and made strides in the field of learning (Gu et al., 2018; Nickel and Kiela, 2017).In comparison to their Euclidean counterparts, these non-Euclidean embeddings have the potential for embeddings that are more expressive. For example, compared to Euclidean embeddings, hyperbolic embeddings exhibit significantly less distortion when representinghierarchical data (Sarkar, 2011). This has led to state-of- the-art outcomes in numerous contemporary applications, including linguistics tasks (Tifrea et al., 2018; Le et al., 2019)
appropriate shape for an input graph. Anintriguing area for future research is the process of selecting or learning the appropriate geometry for a specific discrete graph, even though there are already discrete measures forthe graphs' tree-likeliness, such as Gromov's four-point condition (Jonckheere et al., 2008;Abu-Ata and Dragan, 2016; Chen et al., 2013; Adcock et al., 2013).Assurances based on theory Recent developments in graph embedding model design haveoutperformed state-of-the-art methods in several domains. Nevertheless, our knowledge ofthe theoretical promises and constraints of graph embedding models is currently restricted.Xu et al. (2018), Verma and Zhang (2019), Morris et al. (2019), and Garg et al. (2020) allapply current findings from learning theory to the issue of GRL, which is a new field of study on GNN representational power. If we want to know what the theoretical benefits and drawbacks of graph embedding techniques are, we need to build theoretical frameworks.

References

Written by Feodor F. Dragan and Muad Abu-Ata. Analysis of real-world network structures approximating metric
trees.With reference to: Networks, 2016; 67(1): 49-68.Sami Abu-El-Haija, Rami Al-Rfou, and Brian Perozzi. Improving edge representations using low-rank asymmetric projections. This is the 1787–1796th page of the proceedings from the 2017 ACM Conference on Information and Knowledge Management (CIKM '17). Authors Bryan Perozzi, Alexander A. Alemi, Sami Abu-El-Haija, and Rami Al-Rfou contributed to the work. Use caution: Discovering embeddings of nodes by analysing graphs. This is the 2018 edition of Advances in Neural Information , specifically pages 9180-9190.
The following individuals are taking part: Aram Galstyan, Bryan Harutyunyan, Bryan Perozzi, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Amol Kapoor. By merging sparse neighbourhoods, mixhop constructs higher-order graph convolutional networks. This is the 2019 International Conference on Machine Learning, page numbers 21–29. Written by Feodor F. Dragan and Muad Abu-Ata. Analysis of real-world network structures approximating metric trees.With reference to: Networks, 2016; 67(1): 49-68.Sami Abu-El-Haija, Rami Al-Rfou, and Brian Perozzi. Improving edge representations using low-rank asymmetric projections. This is the 1787–1796th page of the proceedings from the 2017 ACM Conference on Information and

Knowledge
Management

(CIKM

'17). Authors Bryan Perozzi, Alexander A. Alemi, Sami Abu-El-Haija, and Rami Al-Rfou contributed to the work. Use caution: Discovering embeddings of nodes by analysing graphs. This is the 2018 edition of Advances in Neural Information Processing Systems, specifically pages 9180-9190.The following individuals are taking part: Aram Galstyan, Bryan Harutyunyan, Bryan Perozzi, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Amol Kapoor. By merging sparse neighbourhoods, mixhop constructs higher-order graph convolutional networks. This is the 2019 International Conference on Machine Learning, page numbers 21–29. Subjects: Blair
D. Sullivan, Aaron B. Adcock, and Michael W. Mahoney. The structure of large-scale information and social networks resembles a tree. January 2013: Pages 1–10 in Volume 13, Issue 1. Data Mining 2013: IEEE International Conference on.They were Amr Ahmed, Vanja Josifovski, Alexan-der J. Smola, Nino Shervashidze, and Shravan Narayanamurthy. Organic graph factorization on a distributed, enormous scale. Pages 37–48 of the 22nd International Conference on the World Wide Web is where this was included. 2013 (IEEE). Dustin Zelle, Bryan Perozzi, and Rami Al-Rfou were all participants. Learned graphs depict deep divergence graph kernels, abbreviated as Ddgk. Proceedings of the 2019 World Wide Web Consortium Conference on the Internet, 2019. Paolo Mier, Gregorio Alanis-Lobato, and Miguel A. Andrade-Navarro wrote it. How to efficiently use the Laplacian of complex networks to integrate into hyperbolic space? The article was published in Scientific Reports on March 8, 2016. Alves, Luis B. An asynchronous perceptron learning rule with feedback that is combinatorial! Pages 609–618 of Volume 2, First International Conference on Neural Networks Proceedings. IEEE, 1987. These three individuals are Alan Allen, Ivana Balazevic, and Timothy Hospedales "Multi-relational Poincaré graph embeddings" (March 2019), pages 4463–4473, Advances in Neural Information Processing Systems, vol. Among those involved are Matt Lai, Razvan Pascanu, Peter Battaglia, and Danilo Jimenez Rezende. Interconnected structures for learning about things, relationships, and physical phenomena. Advances in Neural Information

Processing Systems, 2016 edition, pages 4502-4510. Graph networks, deep learning, and relational inductive biases are all written by these people: Everyone from Peter W. Battaglia and Jessica B. Hamrick to Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam