Deep Neural Network-based Fusion and Natural Language Processing in Additive Manufacturing for Customer Satisfaction

Abedallah Z. Abualkishik [*], Rasha Almajed

American University in the Emirates, Dubai, UAE
Emails: abedallah.abualkishik@aue.ae; rasha.almajed@aue.ae

Abstract

Modern Machine learning fusion approaches tend to extract features depending on two techniques (hand-crafted feature and representation learning). Hand-crafted features can waste time and are not sufficient for downstream tasks. Unlike representation learning, we automatically learn features with minimum time and effort and are suitable for downstream tasks. In our paper, we provide work on graph neural network methods with details on classical graph embedding approaches and the different methods in neural graph networks such as graph filtering, graph pooling, and the learning parameter for graph following each technique with a general framework or mathematical proof for customer satisfaction. To satisfy customer's feel, this research employs NLP techniques. We describe the adversarial attacks and defenses on graph representation approaches. Also, advanced application of neural graph networks is reviewed, such as combinational optimization, learning program representation, physical system modeling, and natural language processing. Finally, the challenges in geometric neural networks and future research work have been introduced.

Keywords: Machine learning; neural graph networks; graph filtering; graph pooling; optimization; fusion based on NLP; customer satisfaction.

1. Introduction

Real data is variant and can be in many types, such as time series of matrices, etc. Using graphs useful to get the global representation of data. Many applications are suitable for using graphs, such as social networks, traffic, chemistry interactions, etc. Recently, many data types can be converted to graphs [1]. A graph can handle large data in a small set of computational tasks. And can explore anomalous nodes and find related genes to diseases that are named node classification (NC). Recommendations, predicting drug side effects, exploring drug interaction, and knowledge graph (KG) completion are named link prediction [2].

Graph neural networks (GNNs) depend on Deep learning (DL) approaches due to their sufficient performance. They are broadly used in many applications. Recursive NNs introduced firstly on directed acyclic graphs [3]. Deep convolutional neural networks (CNNs) [4] are more advanced than GNNs. Though, CNNs can just run on normal Euclidean space like 2D images and 1D texts. This kind of data can be converted to graph attributes. Hence, it is easy to integrate CNNs with graphs. Expanding deep neural network (DNN) to non-Euclidean domains, named geometric DL, rational inductive basis, graph representation (GR) learning, or GNN [2].

Many studies aim to learn representation in graph nodes, edges, or subgraphs by graph embedding (GE), such as DeepWalk [5], which is the first embedding method for GR and applies random walk (RW) to the SkipGram model [6]. Also, some studies, such as node2vec [7] and LINE [8] have become popular. But these approaches can't share parameters among nodes in the encoder, which causes high time complexities. Also, not capable of generalization to new graphs [9].

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

GNN consists of the filter layer, which enhances the vertex features, and the graph subsampling layer, which intends to coarsen the network. Then produce the graph-level representation. The filter layers are divided into spectral and spatial filters. Where the spectral filter has a good theoretical foundation but loads the whole graph into memory and transductive work mode, on the other hand, the spatial filter can handle the big graph with inductive mode. The pooling graph layer is categorized into direct pooling and hierarchical graph pooling.
These approaches and techniques in GNN are shown to be manipulated by adversarial attacks, which cause a trustworthy issue in geometric networks. GNN attacks help to misclassify the model predictions by manipulating edges or nodes. This is considered an essential area of research during the utilization of GNN in many applications. Composite local features combined by a fusion layer have been found to enhance the cross-subject classification of EEG-MI using feature fusion networks. This is accomplished by reducing overfitting and retrieving complicated features from an inter CNN. Using the EEG Motor Movement/Imaging Dataset (eegmmidb), a multi-branched convolutionary neural network called EEGnet Fusion was created and shown to have an accuracy of 84.1% in cross-subject classification. Different nodes in the EEGnet fusion system had the same amount of filters and blocked size as the original EEGnet model but otherwise were identical.

In our paper, we provide a study on GNN provide a mathematical guide for GE methods that are divided into encoder-decoder graphs and multi-rational graphs. Also, we provide a review of GNN methods divided into graph filter (GF), pooling filter, and graph parameter learning. We discuss the adversarial example and defenses in GNNs with the comparison between attacks and defense approaches. We provide the most recent applications of GNN and discuss the most challenging areas and future directions in GNN.
Our contributions are condensed as follows:
New classification. We propose a new classification of GNN methods, which are classified as graph embedding methods, GNN, adversarial attack methods, and defenses methods.
Comprehensive analysis. We provide the most comprehensive overview of most graph techniques for each classification of the graph; we produce complete mathematical descriptions of representative models and general frameworks for each classification and summarize the comparison approaches.
Future works. We produce the mathematical aspects of graphs from various classifications and evaluate the limitations of current approaches to provide recommend eight potential future works in terms of Deeper GNN, Neighbor explosion, Pre-trained GNN, Robustness, Explainable GNN, Transferability of GNN adversarial Attacks, Simple GNN attacks implementations, Node-based defenses.
The following sections are divided into Section 2. Present compassion with other works. Section 3 provides the Preliminaries. Section 4 provides a review of GE methods. Section 5 provides reviews on GNN methods. Section 6 describes the adversarial attacks in GNN. Section 7 describes defenses on GNN. Section 8 produces the advanced applications in GNN and a case study of customer satisfaction. Section 9 provides the conclusion.

2. Comparison with other works

There are many existing comprehensive surveys on GNN which will be presented in Table 1.

Table 1: Summarization of different surveys in GNN

Study Ref.	Descriptions
[10]	Present a survey on GNN methods in a mathematical fashion. But there are no comparisons between methods and studies.
[11]	Present a survey on CCN only.
[12]	Classify the GNN into four categories of methods: recurrent graph neural networks, Graph convolution networks (GCN), graph autoencoders, and spatial-temporal graph neural networks
[13]	Depending on encoder-decoder as graph embedding technique.
[14, 15]	Specific surveys on adversarial graph examples and their defenses
[16]	Concentrating on GNN have multi-types of vertices and edges(heterogeneous).
[17]	Concentrate on GE approaches in Np-hard problem (combinatorial optimization)

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

[18]	The presence of different graph-based DL approaches
[19]	Focus on the graph attention model.
[20]	A survey on dynamic graphs.

Our paper differs from these studies as we provide a different classification for GNN approaches. The GNN methods are classified as GE methods and robustness methods that portion two adversarial attacks and defenses methods supported with the general mathematical framework for each classification. Also, we present the compassion between methods in GNNs. Furthermore, we introduce the most recent application in GNNs, such as combinatorial optimization, learning programming representation, physical system modeling, and NLP.

3. Preliminaries

Definition 1 (Graph): This is formulated as $G = \{\mathcal{V},\}$ E, V describes a set of N vertex, and $|\mathcal{V}| = N$, $\mathcal{E}$ describes a set of edges linking these vertices.
Nodes or vertices are the main components in the graph where $\mathcal{V} = \{\mathcal{V}_1, \ldots, \}$ and VN. Such as, in social media, users are denoted as vertices. The number of nodes in graphs represents its size, $N = |\mathcal{V}|$. Also, the edge $\mathcal{E}$ is a set of M edges where $\mathcal{E} = \{e_1, \ldots, e_M\}$. The set of edges links between two nodes as $(\mathcal{V}_e^1, \mathcal{V}_e^2)$ [12].
Definition 2 (Directed Graph and undirected graph): When edges in a graph are directed from one node to another is named a directed graph. On the other hand, the inverse directions between two linked edges are called an undirected graph. The undirected graph is considered an individual case from the directed graph. The symmetric adjacency matrix (AM) refers to an undirected graph case [12].
Definition 3 (Homogeneous/heterogeneous): Homogeneous graphs refer to similarities in all vertices and edges in a graph, opposite to heterogeneous graphs, which contain many types of vertices and edges in the graph. A KG is a typical directed heterogeneous graph [12].
Definition 4 (Adjacency matrix)(AM): which is W is an n × n matrix with $W_{ij} = 1$ if $e_{ij} \in E$ and $W_{ij} = 0$ if $e_{ij} \notin E$ [12].

4. Graph embedding

GE is the method that is used to convert nodes, edges, and features into low-dimensional vector space. GE aims to maximally preserve characteristics such as graph structure and information. The data type in the graph is discrete, opposite to the machine learning algorithm that deals with continuous data. Many tasks can perform on features in latent space, such as (edge prediction, NC, etc.). In this part, we present the most general and popular GE techniques. In this section, the GE approaches are classified as shown in Figure 1 [12].

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

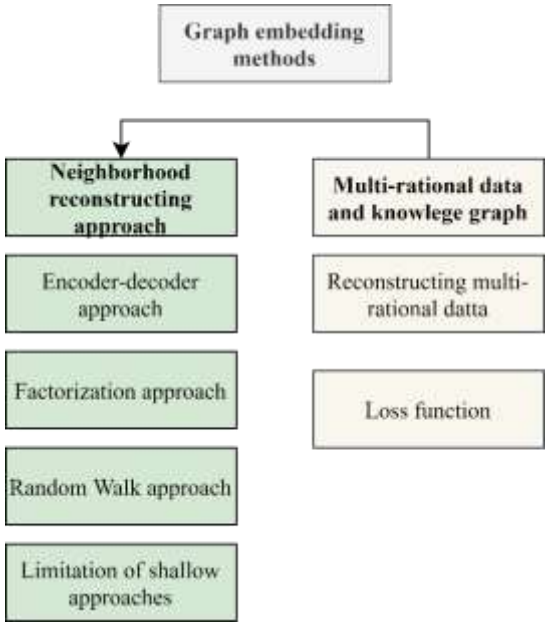

Figure 1: classification of graph embedding

4.1. Neighborhood reconstruction methods

In this part, we will study the GE methods for simple and weighted graphs. These approaches aim to encode nodes in low- dimensional space where geometric relations showed in latent space[21]. As shown in Figure 2.

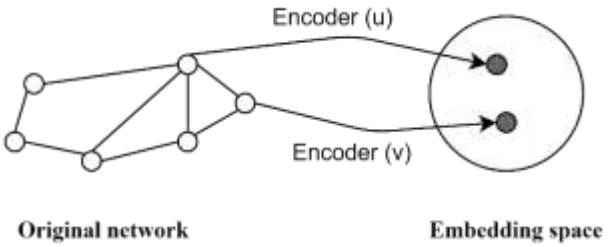

Figure 2: The embedding framework

A) Encoder-decoder approaches

The approaches in this section depend on encoding and decoding the graphs. The encoder aims to embed the node in latent space while the decoder uses the embedded node to reconstruct information about the neighborhood in the original graph, as shown in Figure 3 [21].

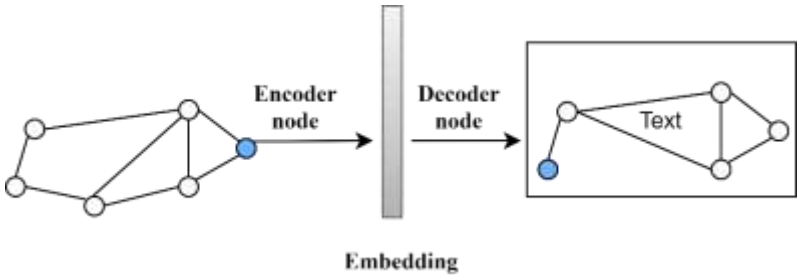

Figure 3: Encoder-decoder framework

The encoder function aims to map the nodes $v \epsilon \mathcal{V}$ to vector space where $X_v \in \mathbb{R}^d$, which is formulated as follows:

$$ENC = \mathcal{V} \rightarrow \mathbb{R}^d \quad (1)$$

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

The encoder function depends on shallow embedding, whereas the encoder function embedding search relies on the node IDs.

The decoder's main aim is to reconstruct the set of neighborhood nodes $S(u)$ or its row $T[u]$ in the original graph from embedded node X_u of node u by the ENC, so the decoder can predict u'. The main implementation can define pairwise decoders as follows:

$$DEC: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+ \quad (2)$$

Prediction of the relationship between nodes (u, v) is the aim of the pairwise decoder. The encoder optimization can reduce the reconstruction loss as follows:

$$DEC(ENC(u), ENC(v)) = DEC(X_u, X_v) \approx SIM[u, v] \quad (3)$$

Where SIM is the similarity between nodes.

Optimization of encoder-decoder aims to attain the reconstruction objective as in Equation (3) by minimizing the loss $\mathcal{L}$ of the training pair node set N:

$$\mathcal{L} = \sum_{(u,v)\in N} \ell(DEC(X_u, X_v), SIM[u, v]) \quad (4)$$

Where ℓ is the loss is measuring function $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ between the decode similarity values X_u, X_v and the true value $SIM[u, v]$. Loss can be minimized using stochastic gradient descent or other optimization approaches. Many encoder-decoder methods are used for GE based on node embedding, which is summarized in Table 2 [21].

Table 2: Summarization of encoder-decoder methods

Approach	Type	Decoder	SIM	Loss function
Laplacian Eigenmaps (LE)	MF	$\|X_u - X_v\|_2^2$	General	$DEC(X_u, X_v), SIM[u, v]$
Graph factorization (GF)	MF (Inner product)	$X_u{}^\top X_v$	$T[u, v]$	$\|DEC(X_u, X_v) - SIM[u, v]\|_2^2$
GraRep	MF (Inner product)	$X_u{}^\top X_v$	$T[u, v], \ldots, T^k[u, v]$	$\|DEC(X_u, X_v) - SIM[u, v]\|_2^2$
HOPE	MF (Inner product)	$X_u{}^\top X_v$	general	$\|DEC(X_u, X_v) - SIM[u, v]\|_2^2$
DeepWalk	RW	$\frac{e^{X_u{}^\top X_v}}{\sum_{k\in v} e^{X_u{}^\top X_k}}$	$P(v\|u)$	$-SIM[u, v] \log(DEC(X_u, X_v))$
Node2vec	RW	$\frac{e^{X_u{}^\top X_v}}{\sum_{k\in v} e^{X_u{}^\top X_k}}$	$P(v\|u)$(biased)	$-SIM[u, v] \log(DEC(X_u, X_v))$

B) Factorization approaches

The matrix factorization (MF) viewpoint in encoder-decoder approaches aims to discover vector space estimate of node-node similarity matrix SIM where SIM generalizes the AM and takes some declared notion of node-node similarity.

Laplacian eigenmaps (LE) are the most popular MF approaches. The decoder based on the L2 distance between the node in vector space is represented as follows:

$$DEC(X_u, X_v) = \|X_u - X_v\|_2^2 \quad (5)$$

Corresponding to the pair nodes' weights, the loss function is defined as follows:

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

$$\mathcal{L} = \sum_{(u,v)\in N} DEC(X_u, X_v), SIM[u,v] \tag{6}$$

Inner-product methods depending on pair embedding nodes, the decoder is represented as follows:

$$DEC(X_u, X_v) = X_u{}^{\top} X_v \tag{7}$$

The Mean-square error is defined as:

$$\mathcal{L} = \sum_{(u,v)\in N} \|DEC(X_u, X_v) - SIM[u,v]\|_2^2 \tag{8}$$

Graph factorization (GF), HOPE, and GraRep are all examples of MF because it optimizes loss function roughly using the following formula:

$$\mathcal{L} \approx \|X^{\top} X - SIM\|_2^2 \tag{9}$$

The main aim of these approaches is the inner product by estimating the node similarity of learned embedding vectors. So, GF depends on AM to define the node similarity. Also, GraRep declares higher order of AM depending on many powers of it. HOPE relies on Jaccard neighborhood overlap to find general similarity measures.

C) Random walk approaches

This approach's goal is to apply stochastic measures of node similarity to get better performance. DeepWalk, LINE, and node2vec are examples of RW approaches.

DeepWalk and node2vec are inner product approaches. They differ in node similarity definition and reconstruction of the neighborhood by optimizing embeddings to encode the statistics of RW. Which roughly learn to embed using the following formula:

$$DEC(X_u, X_v) \triangleq \frac{e^{X_u{}^{\top} X_v}}{\sum_{k\in v} e^{X_u{}^{\top} X_k}} \approx P, T(v|u) \tag{10}$$

Where $P, T(v|u)$ is the probability of going to v on T length RW starting at u, and T ranging from $\{2, \dots, 10\}$. Equation (10) differs from Equation (9) in that Equation (10) depends on stochastic and asymmetric.

Reducing the cross-entropy loss is performed using the following formula:

$$\mathcal{L} = \sum_{(u,v)\in N} -\log(DEC(X_u, X_v)) \tag{11}$$

Where N is the training set of RW that begins from each node, but evaluating Equation (11) is high time complexity $O(v)$. So different optimization approaches are used in DeepWalk and node2vec to reduce the loss. DeepWalk relies on implementing a hierarchical softmax approach which speeds up the computation. Also, node2vec depends on the contrastive noise method to approximate Equation (11) and uses negative samples to estimate the normalizing factor.

Large-scale information network embeddings (LINE) are narrow embedding approaches doesn't clearly depend on RW. LINE Based on optimizing first-order and second-order node co-occurrence. The first order relies on the sigmoid function and SIM of AM as follows:

$$DEC(X_u, X_v) = \frac{1}{1 + e^{-X_u{}^{\top} X_k}} \tag{12}$$

Many studies perform modifications on RW approaches, such as applying the hyperbolic in node2vec inner product instead of Euclidean distance. Another modification on DeepWalk that overlap multiple nodes at each step, the implementation shows similar results to GraRep [22].

D) Limitation of Shallow approaches

Encoder shortage of parameter sharing between nodes which is statistically and computationally inadequate. Also fails to preserve another node attribute to indicate more informative nodes. Another limitation is that shallow GE approaches are inherently transductive, so GE can just insert nodes in the training phase but can't produce the unseen nodes, which represents a problem during big and huge graphs, so memory can't store all graphs.

4.2. Multi-rational data and knowledge graph

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

The key aim of this section is to define the rational graph embedding approaches. KG are represented as tuples $G = (v, \varepsilon, \mathrm{C})$ where v is the nodes, ε *is the edges and* C is the relation between nodes. The multi-rational graphs are also the KG. In this section, we will represent most techniques in multi-rational graphs.

A) Reconstructing multi-national data

Reconstruction can embed multi-rational graphs. If the embeddings X_u and X_v of two nodes, to reconstruct the relation among these nodes. The complication can handle multiple and various variations of edges. The decoder can deal will a rational pair of nodes as follows:

$$DEC: \mathbb{R}^d \times \mathrm{C} \times \mathbb{R}^d \rightarrow \mathbb{R}^+ \tag{13}$$

One of the most known examples of Multi-rational graphs is the RESCAL method [23]. The decoder of RESCAL is represented as follows:

$$DEC(u, \tau, v) = X_u{}^{\intercal} \mathrm{C}_\tau X_v \tag{14}$$

Where $\mathrm{C}_\tau \in \mathbb{R}^{d \times d}$ is a learnable matrix specific to relation $\tau \in \mathrm{C}$ .then train the latent vector X and $\mathrm{C}_\tau \forall_\tau \in \mathrm{C}$ depending on reconstruction loss.

$$\mathcal{L} = \sum_{u \epsilon \mathcal{V}} \sum_{v \epsilon \mathcal{V}} \sum_{\tau \in \mathrm{C}} \| DEC(u, \tau, v) - \mathcal{A}[u, \tau, v] \|^2 \tag{15}$$

Where $\mathcal{A} \in \mathbb{R}^{|v| \times |\mathrm{C}| \times |v|}$ is adjancecy tensor for KG

B) Loss function

Many methods can reduce the loss in Equation (15), such as cross-entropy with the negative sample and max-margin loss.

Cross entropy with the negative sample: presenting cross-function with negative sample help improve performance as follows:

$$\mathcal{L} = \sum_{(u,\tau,v) \in \varepsilon} -\log(\sigma(DEC(X_u, \tau, X_v))) - \gamma \mathbb{E}_{v_n \sim P_{n.u}(v)} [\log(\sigma(DEC(X_u, \tau, X_v))] \tag{16}$$

Where σ represents the logistic function, $P_{n.u}(v)$ indicates the negative sampling distribution over the set of nodes v that, based on u and $\gamma > 0$ is a hyperparameter. Then the output of the decoder is obtained by logistic regression and normalization range from [0, 1] to inferred probabilities. The Mont Carlo approximation can be used to minimize the loss. Negative sampling can increase the performance of the latent vector. Uniform distribution is the most popular way on graph nodes, but it will get a false negative example in the cross-entropy. Some types of negative samples can generate more difficult negative sampling.

Max-margin loss: another important method in multi-rational embedding nodes which can be represented as follows:

$$\mathcal{L} = \sum_{(u,\tau,v) \in \varepsilon} \sum_{v_n \in P_{n.u}} max(0, -DEC(X_u, \tau, X_v) + DEC(X_u, \tau, X_{v_n}) + \Delta) \tag{17}$$

This method uses a contrastive estimation approach. In Equation (17), the decoder output is directly compared, so if the true pair is larger than the negative pair, then the loss is small. The Δ term is named the margin, and the loss will be 0 if the variation in scores is at least that larger than all examples, which are named hinge loss. The most decoder approach is summarized in Table 3.

Table 3: Most decoders in the multi-rational graphs

Method	Score function (DEC)	Relation parameter
RESCAL	$X_u{}^{\intercal} \mathrm{C}_\tau X_v$	$\mathrm{C}_\tau \in \mathbb{R}^{d \times d}$
Rotate	$-\|X_u \circ c_\tau - X_v\|$	$c_\tau \in \mathbb{C}^d$
DistMult	$< X_u, c_\tau, X_v >$	$c_\tau \in \mathbb{R}^d$

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

TransE	$-\|X_u + c_\tau - X_v\|$	$c_\tau \in \mathbb{R}^d$
TransX	$-\|g_{1,\tau}(X_u) + c_\tau - g_{2,\tau}(X_v)\|$	$c_\tau \in \mathbb{R}^d, g_{1,\tau}, g_{2,\tau} \in \mathbb{R}^d \to \mathbb{R}^d$
ComplEx	$Re(< X_u, c_\tau, \bar{X}_v >)$	$c_\tau \in \mathbb{C}^d$

5. Graph neural network

GNN is the group of approaches that seeks to utilize DNN to graph data. The traditional DNN can't deal with graph structure data as the graph data can't deal with grid data.

5.1. General Graph neural network framework

In this section, an introduction to the GNN framework will be introduced. The node-based framework aims to learn the vertex features. The graph-based framework aims to learn the graph features while the node feature is the middle step.

This process can be described as follows:

$$F^{(of)} = h(J, F^{(if)}) \tag{18}$$

Where $J \in \mathbb{R}^{N\times N}$ represent the AM, $F^{(if)} \in \mathbb{R}^{N\times d_{if}}$ and $F^{(of)} \in \mathbb{R}^{N\times d_{of}}$ represent the input and the output feature matrices, d_{if} and d_{of} are their dimensions. If and of refer to input and output filtering. The operator h (·, ·) is named a GF. The GF must only improve the formation of the nodes and not affect the graph structure.

For a node-based framework, the GF process is enough, and the multi-GF process is stacked sequentially to produce terminated node features. Many other processes are important in a graph-based framework to produce the features for the whole graph from the vertex's features. Like the traditional Convolution neural network (CNN), the pooling process aims to condense vertex features to produce graph-level features. Traditional CNNs are added to grid data. The GNN requires a dedicated pooling process. The main aim of pooling processes is to employ the graph structure information to handle the pooling process. The graph can be input for the pooling process and then create a graph with fewer nodes. This operation describes as follows:

$$J^{(po)}, F^{(po)} = pool(J^{(ip)}, F^{(ip)}) \tag{19}$$

Where $J^{(ip)} \in \mathbb{R}^{N_{ip}\times N_{ip}}$, $F^{(ip)} \in \mathbb{R}^{N_{ip}\times d_{ip}}$, and $J^{(ip)} \in \mathbb{R}^{N_{op}\times N_{op}}$, $F^{(op)} \in \mathbb{R}^{N_{op}\times d_{op}}$ are the AM and feature matrices before and after the subsampling process, respectively. Ip and op denote the input and output of subsampling receptively. Where N_{op} represents the number of vertices in the coarsened graph and $N_{op} < N_{ip}$.

5.2. Graph filters

Graph filters (GF) are classified into two filters. The first is spatial-GF, and the second is spectral-GF. The spatial GF explicitly controls the graph structure to obtain the feature enhancement operation in the graph domain. On the opposite, the spectral-GF design the filtering process in the spectral domain.

A) Spectral-GF

In the beginning, Graph Fourier Transform (GFT) uses the following formula:

$$\hat{t} = U^{\intercal} t \tag{20}$$

where U is the LE matrix of G and $\hat{t}$ is the achieved GF coefficients for the signal t. To adjust the frequencies of the signal t, GF coefficients are applied as follows:

$$\hat{t}[i] = \hat{t}[i].\gamma(\lambda_i) \; for \; i = 1,..N \tag{21}$$

where $\gamma(\lambda_i)$ is a function with the frequency λ_i as input which defines the way that the related frequency element should be modulated. Which can present as a matrix:

$$\hat{t}' = \gamma(\Lambda).\hat{f} == \gamma(\Lambda).U^{\intercal} t \tag{22}$$

where Λ is a diagonal matrix involving the frequencies. Then using the Inverse Graph Fourier Transform (IGFT) as follows for reconstruction:

$$t' = U\hat{t}' = U.\gamma(\Lambda).U^{\intercal} t \tag{23}$$

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

where t' is the achieved GF signal. This operation can be viewed as utilizing the operator $U.\gamma(\Lambda).U^{\mathrm{T}}$ to the input graph signal. We provide the most spectral filter graph method by comparing their time complexity in Table 4.

Table 4: Summarization of all spectral filter graph methods.

Method	Input	Time complexity
ChebNet [24]	AM, graph feature matrix (GFM)	$O(m)$
GCN [25]	AM, GFM	$O(m)$
CayleyNet [26]	AM, GFM	$O(m)$
AGCN [27]	AM, GFM	$O(n^2)$
DualGCN [28]	AM, GFM	$O(m)$

B) Spatial-GF

Equivalent to the convolutional process in the image, spatial graph approaches represent convolutions that rely on spatial relations between nodes. Every pixel is linked to its neighboring pixels. The spatial-GF act as in an image, so it updates the main node by convolving the main nodes with its neighbors' representations. The filter output can be denoted as T'. The filtering operation for the node v_i can be represented as:

$$T_i' = \sum_{v_i \in \mathcal{N}(v_i)} g(l_i T_j l_j) \tag{24}$$

Where g() is a feedforward parametric function, node v_i contains a 1-hop neighbor. The l_i is node label information. We summarize the most spatial-GF methods in Table 5.

Table 5: Summarization of all spatial filter graph methods.

Method	Input	Time complexity
MPNN [29]	AM, GFM, The edge feature matrix (EFM)	$O(m)$
graphs [30]	AM, GFM	-
GAT [31]	AM, GFM	$O(m)$
MoNet [32]	AM, GFM	$O(m)$
LGCN [33]	AM, GFM	-
PGC-DGCNN [34]	AM, GFM	$O(n^3)$
CGMM [35]	AM, GFM, EFM	-
GAAN [36]	AM, GFM	$O(m)$
FastGCN [37]	AM, GFM	-
StoGCN [38]	AM, GFM	-
Huang et al. [39]	AM, GFM	-
DGCNN [40]	AM, GFM	$O(m)$
DiffPool [41]	AM, GFM	$O(n^2)$
GeniePath [42]	AM, GFM	$O(m)$
DGI [43]	AM, GFM	$O(m)$
GIN [44]	AM, GFM	$O(m)$
ClusterGCN [45]	AM, GFM	-

5.3. Graph pooling

In image processing, the subsampling layer aims to generalize features. Complex big graphs normally deal with hierarchical structures that are efficient for NC and GC tasks. Many studies concentrate on constructing hierarchical subsampling layers on graphs. In this part, two types of pooling modules will be presented (direct and hierarchical pooling modules) [9].

A) Direct pooling module by choosing different nodes the pooling modules directly learns representation for graph-level from nodes, which are also named as readout functions.

Simple Node Pooling. The common method that implements node-wise operations is to obtain global representation for node features.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

Set2set. Implemented by MPNN as the readout function to find GR. This method aims to handle the disordered set $T = \{(h_v^T, x_v)\}$ depending on the LSTM approach to present an unvarying ordered representation after predetermining moves numbers.
SortPooling. [40] aims to sort the latent node according to the node structural roles and input it into CNN's to obtain the representation.

B) Hierarchical pooling module, the previous approaches directly learn GR from vertices, and they do not focus on the hierarchical structure of the graph. Many studies concentrate learn GR through hierarchical pooling.
Graph Coarsening. Many approaches depend on this approach, such as spectral clustering algorithms, which have bad performance because of the Eigen decomposition step. Also, Graclus provided a speed-up for node clustering and presented it as a pooling module. Also, studies such as MoNet and ChebNet are based on the Graclus approach to integrating pair nodes with extra nodes to guarantee the pooling process types a balanced binary tree.
Edge-Conditioned Convolution (ECC) is based on a recursive down-sampling operation that relies on dividing the graph into two parts by the sign of the biggest LE.
DiffPool. [41] uses a learnable hierarchical clustering module by training a linear responsibility matrix P^t for every layer:

$$P^t = softmax\left(GNN_{t,pool}(C^t, F^t)\right), \quad C^{t+1} = (P^t)^T C^t P^t \tag{25}$$

Where Fits the vertex feature matrix and $C^t$is toughened AM of layer t. P^t represents the probabilities that vertices in layer t can be denoted to a coarser node in layer $t+1$.
gPool. obtain projection score based on the projected vector in every vertex and choose the vertex with top-k scores. Contrasted to DiffPool, this approach obtains a vector rather than a matrix for every layer, which minimizes the time complexity. But doesn't focus on graph structure on projection operation.
EigenPooling. based on vertex features and local structure together. The local GFT utilizes to get sub-graph data, but it is insufficient because of graph eigendecomposition.
SAGPool. self-attention approach, which lets the downsampling approach focus on both features and topology. This is the approach in sufficient time and space complexity.

5.4. Learning parameters

In this part, an introduction to node and graph classification (GC) will be presented to explain the way parameters learn in GNN.
Node classification parameters if node v can split into disjoint v_l and label v_u.
The aim of NC is to discover the labeled vertex v_l to predict the labels of v_u. The representation can be produced using the geometric neural network. This representation is obtained to train a vertex classifier. Specifically, let $GNN_{node}(,)$ represent a GNN model with many stacked filter layers. This function input and output are present in the graph structure, and the vertex features are as follows:

$$S^{(out)} = GNN_{node}(W, S; \theta_1) \tag{26}$$

Where $\theta_1$is the model parameter, $W \in \mathbb{R}^{N\times N}$ is AM, $S \in \mathbb{R}^{N\times d_{in}}$ is the input feature, and $S \in \mathbb{R}^{N\times d_{out}}$ is the output feature. So, NC can be represented as follows:

$$Z = softmax(S^{(out)}\theta_2) \tag{27}$$

where $Z \in \mathbb{R}^{N\times C}$ is the output logits for nodes, $\theta_2 \in \mathbb{R}^{d_{out}\times C}$ is the parameter matrix to convert the features $S^{(out)}$ into the dimension equal to classes C amounts. this operation can present as follows:

$$Z = f_{GNN}(W, S; \theta) \tag{28}$$

Where f_{GNN} contains all functions in Equations (26) and (27). And the θcan be reduced as follows:

$$\mathcal{L}_{train} = \sum_{v_i \in vV_l} \ell(f_{GNN}(W, S; \theta)_i, y_i) \tag{29}$$

Where $f_{GNN}(W, S; \theta)_i$ represent the row number i of the output, and $\ell(,)$ is a loss function.
Graph classification parameters

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

Every graph is handled as an instance with a linked label. The training data is represented as $T = \{g_i, v_i\}$, where v_i is the label of graph g_i. The classification process of the graph is to model the training stage on T to get sufficient predictions for unlabeled graphs. The GNN model is normally denoted as a feature encoder that gets representation from input features as follows:

$$S_g = GNN_{graph}(g, \theta_1) \tag{30}$$

Where GNN_{graph} is the GNN model that learns representation which involves GF and pooling layers. This graph-level representation is used to obtain the GC as:

$$z_g = softmax(S_g \theta_2) \tag{31}$$

where $\theta_2 \in \mathbb{R}^{d_{out} \times c}$ converts the GR to the dimension number of labels C and $z_g \in \mathbb{R}^{1 \times c}$ represented the predicted logits for g. this process can present as follows:

$$\mathcal{L}_{train} = \sum_{g \in T} \ell(f_{GNN}(g_i, \theta), y_i) \tag{32}$$

6. Adversarial attacks on Graph Neural Networks

The classical DL model can be tricked by adversarial attacks by adding some perturbation in the training or testing stage and causing misleading for the model. Also, the graph can be attacked by perturbing the graph structure or vertices features to trick the GNN models, as shown in Figure 4, Where node d in the original graph is supposed to be labeled as the colored label. But the attacker aims to manipulate and modify the structure and graph edges which leads to misleading labeling.

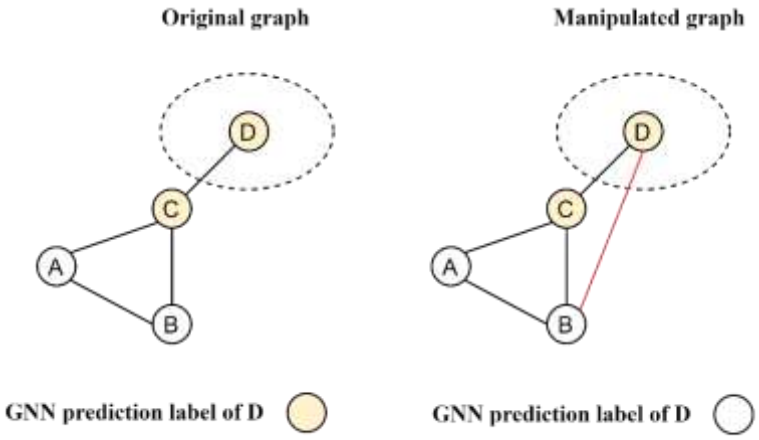

Figure 4: Adversarial attacks in GNN

6.1. The general formula of adversarial attacks

Based on the graph definition in the preliminaries section. We can describe a general formula of attack aims, which seeks to distort the target model by reducing the loss of attacks. That can be represented as follows.

If $G = (W, M)$ where W is AM and A is the attribute matrix of the node. And the target node subset $V_t \subseteq V$. Let C_u describe the label for node u. The attacker aims to discover the attacked graph $\hat{G} = (\widehat{W}, \widehat{M})$ that reduces the objective of attacks as follows:

$$min \mathcal{L}_{attack}(f_\theta(\hat{G})) = \sum_{u \in V_t} \ell_{attack}(f_\theta * (\hat{G})_u, y_u) \tag{33}$$

$$s.t., \quad \theta^* = arg\ min_\theta \mathcal{L}_{train}(f_\theta(G'))$$

Where $\mathcal{L}_{attack}$ is the attack loss function, and G'may is G or $\hat{G}$. where $\hat{G}$ is selected from $\varphi(G)$ constrained domain which is typically used as:

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

$$\|\hat{w} - w\|_0 - \left\|\hat{M} - M\right\|_0 \leq \Delta \quad (34)$$

Where Δ is the cost of fixed perturbation? The graph-level attacks can be described similarly, but most attacks and defenses targets node-level.

6.2. Types of attacks on GNN

A) Attack timing

The attacker attack model during the training phase or testing phase, such as an evasion attack performed in the testing phase. A poisoning attack happens before the model training phase by adding perturbation to the data, which leads to misleading the model.

B) Nature of perturbation

The perturbations can be added to the GNN model in different ways. Such as altering node features, inserting/removing edges, and inserting false nodes. The perturbation must be unseen, so it can't be discovered. These perturbations are shown in Table 6.

Table 6: Summary of perturbations nature

Altering Feature	inserting or removing Edges	Inserting false nodes
An adversary can somewhat modify the node features during the graph structure building.	The adversary can insert or remove edges for the existing nodes under a specific budget of overall events.	The adversary can add false nodes and integrate them with the original nodes in GNN.

C) Adversary Goals

Targeted attacks: The adversary seeks to mislead the model depending on a few testing data based on $V_t \subseteq V$. These kinds of attacks can be directed where the adversary performs manipulation on the features or edges of the target nodes or just affect the target by manipulating the other manipulation.

Untargeted Attack: The adversary seeks to add perturbations to the training model, which leads to misleading the testing phase. Which describe as follows $V_t = V$.

D) Adversary Information

The attacker's information refers to the amount the attacker knows about the mode. Generally, there are three situations:

White-box Attack: The adversary knows the model parameter, inputs, and classes.

Gray-box Attack: The adversary has some information about the target model. The attacker uses the known information to get the other or mislead the model.

Black-box Attack: The adversary doesn't know model parameters or labels. The attacker depends on querying to get information.

E) Types of attacked models:

In this part, we will review most models that are shown to be vulnerable to adversarial attacks. The most famous type of geomatical neural network is GCN which is a type of graph learning approach. This approach depends on aggregating and passing the knowledge from its neighbor nodes which leads to reaching superior performance. But it has been shown that GCN is susceptible to the adversarial example. Also, some studies showed that many other geometric neural networks, such as graph attention networks (GAT), graph column networks (GLN), or JKNet are susceptible to adverbial attacks. Also, GE approaches such as Deepwalk and node2vec. We summarized the most attack models in Table 7.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

Table 7: Summarization of GNN attacks (information: W = White-box, B=Black-box, G= Gray-box; Goals: Target=T, Untargeted=U)

Approach	Information	Goals	Timing	Perturbation nature	Target model	Task
PGD, Min-max	W	U	Both	Inserting or removing Edges	NC	GNN
IG-FGSM IG-JSMA	W	Both	Test	Insert/ remove Edges, Altering feature	NC	GNN
Wang et al.	W, G	T	Train	Insert/ remove Edges	NC	CNN
Nettack	G	T	Both	Insert/ remove Edges, Altering feature	NC	GNN
Metattack	G	U	Train	Insert/ remove Edges	NC	GNN
NIPA	G	U	Train	Inserting false node	NC	GNN
RL-S2V	B	T	Test	Insert/ remove Edges	Graph and NC	GNN
ReWatt	B	U	Test	Insert/ remove Edges	GC	GNN
Liu et al.	W, G	U	Train	Flip label Altering features	Classification Regression	Semi-supervised graph
FGA	W	T	Both	Insert/ remove Edges	NC, Community Detection	GE
GF-Attack	B	T	Test	Insert/ remove Edges	NC	GE
Bojchevski et al.	B	Both	Train	Insert/ remove Edges	NC, Community Detection	GE
Zhang et al.	W	T	Train	Inserting or removing facts	Plausibility Prediction	GE
CD-Attack	B	T	Train	Insert/ remove Edges	Community Detection	Community detection algorithm

7. Defenses against graph neural network attacks

In this section, we provide a review of most defense approaches to geometric neural networks.

7.1. Graph purification training

This technique aims to achieve model robustness by adding some perturbations to training data. There are generally two phases in adversarial training: 1) producing perturbations and 2) training the model with these perturbations. The defense function can be represented as:

$$min_\theta max_{\delta_J \in P_J,\ \delta_y \in P_y} \mathcal{L}_{Training}(f_\theta(J + \delta_J, y + \delta_y)) \quad (35)$$

Where the min-max problem has δ_J, δ_y is the perturbation of J, y represents AM and feature matrix, respectively, and P_J and P_y are the domain of perturbation.

This technique can be done on edge perturbation, node perturbation, and both node and edge perturbations.

Edge perturbations. This is based on perturbation that aims to randomly fall edges. This can enhance the robustness of the model in tasks such as node or GC. Xu et al. produce an attack that targets the topology depending on the PGD attack. Another contribution aims to find perturbations in an unsupervised model by increasing the effect of random noises in the low-dimensional space. This technique increases robustness in NC in

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

DeepWalk. For edge prediction tasks based on modeling the interactions between analyst and adversary, such as in the Stackelberg game, which keeps removing the remaining edges.

Node Perturbations perturbation of the vertex feature help to impose the smoothness between the original vertex and the example. Feng et al. depend on a dynamic regularize technique to push the model to learn to stop graph examples. Another contribution is the concentration on the earliest hidden layer of geometric models to constantly perturb the AM and feature matrix.

Node-edge Perturbation depending on adversarial examples approaches such as Nettack and CD-Attack, proposes further smoothness defense techniques, which shows enhancement in NC and community detection tasks. Also, some studies depend on reinforcement learning to discover the mixed attacks in both vertex and edges. We summarize the most recent defense method in Table 8.

Table 8: Summarization of GNN defenses

Task	Based-Model	Adversary method	Metric
NC	GNN	Metattack	Accuracy
NC	GCN, t-PINE	LowBlow, Nettack	Right rate
Community recognition	Community recognition algorithms	-	Accuracy
Fraud recognition	Sybil recognition	Alter label, Graph production	AUC
Manipulating opinion	Graph model	-	-
Recommender system	GCN	Random, Mixed, Hate, Average	MAE, RMSE
NC	GNN	Nettack, Meta-self, Random	Accuracy
Fraud recognition	GNN fraud recognition	Singleton, IncDS, Random, IncPR, IncBP	Practical effect
GC	GIN	Graph production	Clean and backdoor accuracies, ASR

7.2. Perturbations Detection

Rather than producing perturbations in the training stage, some techniques aim to discover and minimize the influence of attacks and suppose that data has already been perturbed. The graph structure can be exploited to distinguish between the original data from perturbed ones.

Graph Preprocessing. Xu et al. introduce an approach that discovers possible vulnerable edges depending on producing graphs, edge prediction, and discovering the outliers. Rather than edges, Zang et al. present vertices detection depending on topological attacks and practical analysis of the difference between the proximity distributions of vertices toward their neighbors. For graph structure, depending on the interpretations of adversaries, choose to insert edges over deleting edges, and the edges are usually inserted between different vertices. Another contribution, separate sub-graphs from the perturbed training set and then utilize outlier discovering approaches to find and filter attacked edges. All these approaches were performed before training normal GNN models.

Graph Training. Instead of finding the perturbed vertices or edges before the training stage, many studies aim to implement attention techniques for dynamic discovery and reduce vulnerable data through training. Zhu et al. suggest high estimation uncertainty for perturbed vertices and evaluate the attention weights depending on the vector space variation in GCN gaussian. Another contribution assumes training perturbed GCN on attacked edges produced by Nettack and transfers the capability to allocate slight attention weights to attacked edges depending on meta-learning.

Robustness Certification. Zunger et al. produce robustness certificates to assess the protection of individual vertices under adversarial examples. Further, focus on the structure of the attack, and focus on attribute attacks. Integrate the GNN with these certificates during training, causing a strict protection assurance of more vertices. Some studies depend on the robustness certificate for community recognition approaches under structural attacks. Another contribution shows that the polynomial spectral GF is steady under structural attacks.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

Complex Graphs Instead of homogeneous graphs, a study on the sensitivity of KG edge prediction approaches for perturbed facts and the identification of facts. Another contribution explored the discovery of perturbed vertices in heterogeneous graphs to improve the robustness of Android malware discovery systems.

8. Applications and Case Study

8.1. Combinatorial optimization

Various graph Np-hard problem has no polynomial-time approaches that are obtainable for them and are usually handled by using heuristics methods. But heuristic algorithms need considerable knowledge and much trial and error. The goal of GNN is to discover these heuristics and find solutions for unseen tasks.

Minimal Vertex Cover (MVC): From the graph definition in the preliminaries section, the graph output will be vertex subset P of the smallest size such that for all $\{u, v\} \in \mathcal{E}$, at least one of u *or* v is in P. MVC aims to detect the VC with the minimum number of vertices.

Traveling Salesman Problem (TSP): Based on the graph definition in the preliminaries section, if some routes connect some towns, the TSP aims to discover the smallest path that visits each town one time and turns back to the beginning city. It describes as vertices the towns and edges are the paths linking them. The distance among towns can be represented as weights on the edges.

Max-Cut (MAXCUT). Based on the graph definition in the preliminaries section, a cut $T = \{P, \mathcal{V}/P\}$ is a split of $\mathcal{V}$ into two disjoint subsets P and $\mathcal{V}/P$. Its consequent cut-set is the subset of edges $\varepsilon_T \in \varepsilon$ with one termination point in P and the other termination point in $\mathcal{V}/P$. The dilemma of max-cut is to get such a cut, where cut-set weight $\varepsilon_T$refer to as $\sum_{(u,v)\in\varepsilon_T} W(u, v)$ are maximized where $W(u, v)$ refers to the weight of edge (u, v).

Maximal Independent Set (MIS). If a graph G, an independent set is a subset of vertices P ⊂ V where no pair of vertices is linked by an edge. The dilemma of MIS is to get the independent set with the biggest number of vertices.

These Np-hard issues can be represented as vertices and edge problems, where the aim is to discover if the graph is a solution or not. For example, the MVC problem can be represented as an NC problem. Also, the TSP can be represented as a problem of node choosing or edge annotation. GNNs are appropriate to deal with these issues and provide powerful training examples. Though, directly addressing these issues as simply vertex/edge annotation tasks may cause worthless performance. Such as, in the task of the MIS problem, pair of linked vertices may be annotated as 1 simultaneously with the inference. This can lead to a worthless independent set. Therefore, many heuristic algorithms are normally used with GNN to discover acceptable solutions.

8.2. Learning Program Representations

Recent methods aim for automated source code (SC) to tackle variable abuse and suspicious software detection. Generally, it can be done by treating SC as "articles" in a certain language and applying the NLP approaches. But SC tokenization process is not usually capable of discovering the syntactic and semantic relations in the code. Lately, GNN has been utilized to learn representations to enable downstream tasks. In this part, we will discuss the transformation of SC into a graph. Also, we define the downstream tasks and procedures that GNN can be used to deal with these tasks. There are many techniques for building graphs from SC.

Abstract Syntax Tree (AST). Most public representation graph technique encodes the conceptual syntactical composition of the SC. Syntactic errors can be found using AST to recognize the composition of code. In language programming, the syntax is not the terminal of the program, it is contained in the syntax node in the graph. In contrast to syntax tokens that are represented at the ends of programming. Then directed edges are assumed to find the relations between the child and parents.

Control Flow Graph (CFG). Obtain all possible paths to be overpassed in a program through the implementation. This graph involves statements and conditions as nodes. The conditional statement act as vertices of creating various paths. The edge in this graph denotes makeover control among statements.

Data Flow Graph (DFG). Represent the way of variables utilization across the program. The nodes are referred to as variables, and the edges correspond to variable alteration or access.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

Natural Code Sequence (NCS). It is a procedure of the SC, where the edges link adjacent vertices of code tokens corresponding to the sort in the SC.

8.3. Physical system modeling

Modeling physical systems has become an essential field in GNN, which can represent the objects and interactions between objects. This requires a brief and good understanding of physical law and predictions. The graph represents the objects and interactions as nodes, vertices, and edges, respectively. For example, atoms interactions can act as nodes and edges for atoms and interactions, respectively. Also is the robot with multiple parts that are linked with joints. The bodies and joints can be represented as vertices and edges, respectively. Based on the current state, we can predict the next state of the model [9].

GR has been represented in CommNet to obtain node updates relying on nodes' prior and the average of all nodes' prior representations. Another contribution by Watters et al. introduces VIN that integrates CNNs, RNNs, and interaction networks.

Reasoning GNN on objects, relations, and physics is a sufficient way. Such as, which obtains object trajectory as input and refers to an explicit interaction graph for discovering a dynamic model concurrently. Also, Sanchez et al. introduce a GNN model on a robotic system for graph encoding. Also, the authors improve the stably controlling by integrating GNN with Reinforcement learning.

8.4. Natural language processing and Data fusion

Natural language processing (NLP) is the most popular application for GNN. GNNs refer to document labeling by discovering the relation between wards and documentation. Besides sequential data in NLP but also internal graph structure can exist for syntactic dependency tree (SDT) [12]. An SDT represents the grammatical connections between phrase words. Marcheggiani et al. introduce the lexical GCN depending on SDT sentences to combine hidden word representation. This approach executes over phrase encoder of CNN and RNN. Another study aims to add machine translation to the Syntactic GCN. Further improvement for this model is to deal with the semantic reliance graph of a phrase. Graph2seq learning learns to produce phrases with a similar meaning supplied with a syntactic graph of root words (named Abstract Meaning Representation). Song et al. introduce LSTM- a based graph to encode semantic information at the graph level. Another implementation aims to add GGNN for machine translation based on graph2seq learning. The opposite of seq2graph learning, producing a syntactic graph given a phrase is very valuable in knowledge learning.

In this paper, we offer a unique framework that makes use of DL to glean insights from survey responses made by customers. Deep learning (DL) expands on conventional ANN designs to process unstructured input types including photos and text. Our primary objective is to develop a system capable of translating free-form customer feedback into 11 distinct categories that have been identified as important in determining customer satisfaction.

We provide a four-stage process for assessing customer satisfaction data as a series of activities leading to value generation. The CRISP-DM technique has been extensively utilized, particularly in text mining attempts for customer feedback, and this method is a modified version of that procedure.

A CRISP-DM framework is a generic tool for implementing many kinds of learning algorithms in commercial settings. Business comprehension, data analysis, data preprocessing, modeling, assessment, and implementation are the six main stages. To analyze customer satisfaction data using natural language processing and text mining, the suggested framework modifies this approach. To put it another way, we build upon the CRISP-DM procedures in order to create a model specifically for analyzing the level of pleasure felt by the clientele.

The objective of the framework is to design an automated system capable of tagging free-form customer comments in accordance with the numerous factors that contribute to CSAT. Simultaneously, we attempt to glean as much data as we can from the correlation between free-form comments and star ratings. Word clouds and similar descriptive-analytics technologies may help businesses accomplish this second goal.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

To better understand the software, this approach makes use of not just the score that labels users as promoters, auras, or critics, but also their free-form remarks in response to the accompanying open-ended inquiry. The suggested framework's four stages are summarised in Figure 5.

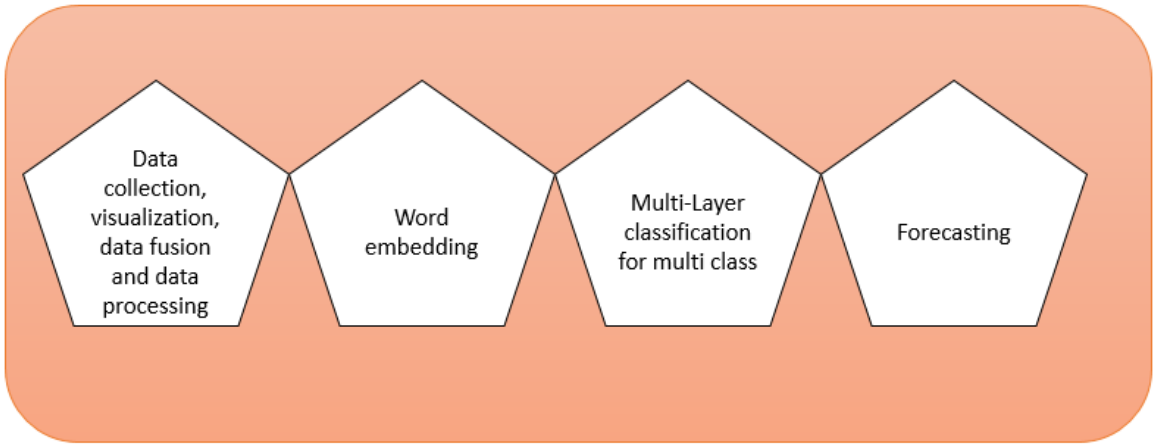

Figure 5: The proposed framework of this paper.

Incorporating Data Visualization and Processing

The first phase includes three individual phases that make up the CRISP-DM procedure. The first subsection of the feature extraction stage is concerned with business comprehension and the establishment of text-mining objectives. The next phases include a more in-depth investigation of one or more of these topics using machine learning, and this step finds the overarching patterns in the data. The outcomes of this phase are the business's goals and measures of success.

Step 2 of the CRISP-DM process is the data analysis and visualization phase. In this phase, you'll do two things: (1) learn about and evaluate the quality of the data used as input to the modeling process, and (2) find some early patterns that are important to grasp the customer's perspective. Word clouds, with phrases and words shaded according to the predominating label (activator, passive, or detractor), are an especially useful tool, in our opinion. A report on the quality of the data and the results of the data discovery process is the product of this stage; it contains the information required for proper data preprocessing.

Notice that the first step determines the drivers that have a significant influence on the NPS score. Managerial insights may be gained from the automated driver identification procedure outlined in Step 3 by prioritizing comments based on the drivers that are part of them, resulting in tailored actions to improve the experience of critical customers.

Data collection, annotating, and cleaning comprise the third substep. The completed product is a collection of cleaned and prepared data for further model training.

We conducted a telephone poll and typed the replies by hand to compile the data. We also manually annotate the free-form answers, creating tags for the machine-learning algorithm at the same time. We use open-ended responses from the telecommunications industry to demonstrate the tagging process (mobile phones). When discussing the driver's "pricing," the sentences "The call quality is bad, and the mobile phone's functions are poor" and "It has cheap fares, and you may chat more for less money" are relevant. For text mining, we used a typical approach to preprocessing data, which involves eliminating stopwords (common words like adjectives and adjectives) and stem (a technique that reduces derivative terms to their root forms).

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

Thirdly, we embed words.
The textual material we need to derive patterns from is unstructured and so must be first organized. By and large, the term frequency (TF) - document term frequency (IDF) method has been used to complete this task, which is known as word embedding. Unique terms in texts are used to create "document vectors" in this method.

A state-of-the-art method considers whole passages, with text preprocessing included right into the learning system. As a DL strategy for semantic embeddings, Deep Bidirectional Portrayals from Transformer (BERT) have been more popular in recent years due to their superior performance compared to other word-embedding techniques. An important technological component of BERT is that it applies the bidirectional training approach of the widely used transformer long short-term memory for translation software to the field of natural language processing. Its success is predicated on its capacity to read a whole string of words simultaneously, picking up on the meaning of individual words based on their context. Unlike traditional word embedding methods, which only read text inputs in one way (often left to right), BERT may read text inputs in both directions simultaneously.

BERT covers 15% of the phrases arbitrarily from the input text to anticipate the masked words using the decoder, so achieving a bidirectional model of phrases. In order to learn the connections between phrases, the BERT system employs a similar method. Whether you provide BERT with a pair of phrases, it will figure out if the second comment is the one that comes next in the text or not. In order to do this, we use inputs consisting of pairs of phrases that appear in the text in the right order 50% of the time and a randomly selected phrase from the text as the two sentences in the other 50% of the time.

Lastly, a mixed word-embedding strategy is investigated. For the pre-trained BERT system, we join the TF-IDF vectors with the original comments. When a client uses terms like "price" or "cheap," these phrases directly correspond to the driver "price" in the TF-IDF framework. Accordingly, the goal is to make the most of both universes by modeling the inter-word interactions in a phrase with BERT and retrieving the details of individual words using the TF-IDF architecture. Based on our research, we were unable to find a straightforward methodology that combines the two methods; as a result, our method represents an original advancement. Other methods, however, combine TF-IDF with BERT to get a better result. A thick vector encoding from Word2vec or worldwide matrices is proposed by Schmidt, for instance, with weights determined by the IDF function (GloVe). We highlight the fact that the uniqueness and originality of our suggestion are generally utilized, despite this little change in methodology.

For the sake of thoroughness, we evaluate BERT (tweaked) against three other approaches: All three of these methods—TF-IDF, pre-trained BERT (without fine-tuning), and the aforementioned hybrid TF-IDF/BERT approach—have their advantages and disadvantages. Figure 6 shows the results of these methods.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

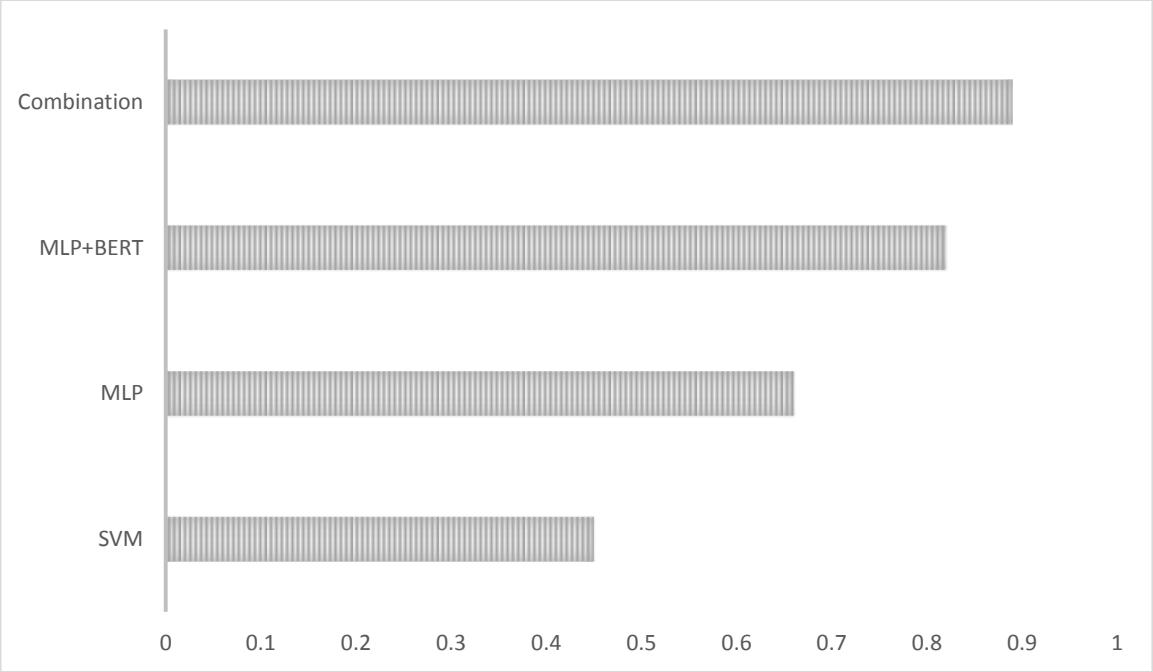

Figure 6: The outcomes of the proposed methods.

Figure 6 reveals several important insights. To start, if we employ TF-IDF as an embedding approach, the MLP network outperforms SVM. That's why this method of categorization is being considered for the remaining tests. Secondly, a meaningful word-based strategy based on AI-learned vectors is shown to be effective when employing pre-trained BERT embedding (from F1 0.45 to F1 0.66). Incorporating the fundamental characteristics from phrases that sum up the TF-IDF architecture, the hybrid BERT/TF-IDF technique only marginally improves generalization ability (from F1 0.66 to F1 0.82). Third, the best results are obtained by adding a multi-label work contribution to the BERT design, which allows us to fine-tune the word-extracted features while the classification model is being trained. The F1 score of 0.89 is a great result for a multi-label clustering algorithm with 11 labels achieved using this method.

9. Conclusion

In this paper, we provide a review article on geometric neural networks. We discussed the graph embedding approaches depending on neighborhood reconstruction methods and multi-relational graph methods. For the neighborhood reconstruction methods, we provide the differences between encoder-decoder, factorization, RW approaches, and the limitation of shallow embedding graphs. On the multi-rational and KG methods, we show the procedure of multi-rational graph reconstruction and how to minimize the loss function. These are followed by a comparison with the most popular graph embedding approaches. Also, the graph neural network was discussed and presented a general framework for it. The graph neural network approaches are divided into graph filter layer, pooling filter layer, and graph parameter learning. Also, we provide a review of the robustness of GNNs, which is important for appending to the geometric neural network. We present the different adversarial example approaches from different perspectives. These approaches show that geometric neural networks are suspectable to small perturbations on graph edges or vertex features. So, we represent the different defense approaches that help to enhance the robustness of graph-based models. We produce a review of the most advanced and important applications in the geometric neural network, such as combinatorial optimization, physical system modeling, learning programming representation, and natural language processing. The challenging and future direction has been shown and discussed for supporting more research efforts.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

References

[1] J. Xu, "Representing Big Data as Networks: New Methods and Insights," University of Notre Dame, 2017.
[2] Y. Ma and J. Tang, *Deep learning on graphs*: Cambridge University Press, 2021.
[3] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, pp. 714-735, 1997.
[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 1998.
[5] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701-710.
[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.
[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855-864.
[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on the world wide web*, 2015, pp. 1067-1077.
[9] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57-81, 2020.
[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, pp. 18-42, 2017.
[11] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, pp. 1-23, 2019.
[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, pp. 4-24, 2020.
[13] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *arXiv preprint arXiv:2005.03675*, 2020.
[14] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, L. He, *et al.*, "Adversarial attack and defense on graph data: A survey," *arXiv preprint arXiv:1812.10528*, 2018.
[15] L. Chen, J. Li, J. Peng, T. Xie, Z. Cao, K. Xu, *et al.*, "A survey of adversarial learning on graphs," *arXiv preprint arXiv:2003.05730*, 2020.
[16] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[17] Y. Peng, B. Choi, and J. Xu, "Graph Learning for Combinatorial Optimization: A Survey of State-of-the-Art," *Data Science and Engineering*, vol. 6, pp. 119-141, 2021.
[18] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[19] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention models in graphs: A survey," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, pp. 1-25, 2019.
[20] Y. Huang, H. Xu, Z. Duan, A. Ren, J. Feng, Q. Zhang, *et al.*, "Modeling Complex Spatial Patterns with Temporal Features via Heterogenous Graph Embedding Networks," *arXiv preprint arXiv:2008.08617*, 2020.
[21] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, pp. 1-159, 2020.
[22] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international conference on information and knowledge management*, 2015, pp. 891-900.
[23] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Icml*, 2011.
[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844-3852, 2016.
[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2016.
[26] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, pp. 97-109, 2018.
[27] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021

[28] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 499-508.
[29] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, 2017, pp. 1263-1272.
[30] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025-1035.
[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2017.
[32] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115-5124.
[33] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1416-1424.
[34] D. V. Tran, N. Navarin, and A. Sperduti, "On filter size in graph convolutional networks," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1534-1541.
[35] D. Bacciu, F. Errica, and A. Micheli, "Contextual graph Markov model: A deep and generative approach to graph processing," in *International Conference on Machine Learning*, 2018, pp. 294-303.
[36] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," in *UAI*, 2018.
[37] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," in *ICLR*, 2018.
[38] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," in *ICML*, 2017.
[39] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *NeurIPS*, 2018.
[40] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[41] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 4805-4815.
[42] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, L. Song, *et al.*, "Geniepath: Graph neural networks with adaptive receptive paths," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 4424-4431.
[43] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2018.
[44] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *ICLR*, 2018.
[45] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257-266.

Doi: https://doi.org/10.54216/FPA.030105
Received: November 11, 2020 Revised: January 05, 2020 Accepted: March 19 2021