# MACHINE LEARNING BASED ON GRAPHS FOR PREDICTIVE MODELING IN INTRICATE NETWORK DATA

Dr. Sandip D. Satav*
Dr. Poonam D Lambhate[1]
Dr. Chandraprabha A Manjare[2]
Dr. Shailesh M Hambarde[3]
Dr. Aparna S Hambarde[4]
Mrs. Aarti S Satav[5]
Associate Professor*, Department of Information Technology, JSCOE
Professor[1], Computer Engineering, JSCOE
Professor[2], Electronics & Telecommunication Engineering, JSCOE
Associate Professor[3], Electronics & Telecommunication Engineering, JSCOE
Assistant Professor[4], Computer Engineering, KJ's COE
Manager[5], SBI, Pune
Pune 411028, India
Manager[5], SBI, Pune
*Corresponding Author

**Abstract**: As contemporary network systems become more intricate and interconnected, machine learning has emerged as a potent instrument for predictive modeling. Because of their capacity to manage complex network data structures, including social networks, biological networks, communication networks, and Internet of Things systems, graph-based machine learning models have become increasingly popular among these. In order to increase prediction accuracy, graph-based techniques offer a framework for illustrating the connections between items in a network. This paper offers a thorough investigation of machine learning methods for predictive modeling in complex networks that make use of graph structures. The study assesses important graph-based algorithms, looks at how they are used in different fields, and talks about the difficulties and potential avenues for further research.

**Keywords:** Graph Neural Networks (GNN), Graph Convolutional Networks (GCN), Internet of Things (IoT), Communication Networks, Graph Attention Networks (GAT).

## 1. INTRODUCTION

Predictive models that can handle, evaluate, and draw conclusions from this complex network data are becoming more and more necessary as a result of the explosion of data produced by interconnected systems, from social networks to Internet of Things (IoT) infrastructures. Network data, as opposed to conventional tabular data, is naturally organized as graphs, with entities (nodes) and their relationships (edges) forming intricate patterns. This structure can be used for a number of predictive tasks, such as link prediction, node categorization, community detection, and anomaly detection, thanks to graph-based machine learning models. A natural method of modeling entities and their relationships is provided by graph-based machine learning, also known as graph machine learning (GML), which applies machine learning algorithms to graph data. Algorithms for predictive modeling on graphs must be able to capture dependencies at both the node and graph levels, going beyond straightforward regression and classification tasks. Because real-world networks are so common, it is essential to comprehend how to apply graph-based techniques for predictive modeling in order to advance machine learning applications in complex networks. Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), Graph Auto encoders (GAEs), and other fundamental ideas of graph-based machine learning are covered in this study. We emphasize their uses, performance assessment, and deployment issues across several areas.

## 2. CONTEXT AND RELATED RESEARCH

In order to overcome the shortcomings of conventional machine learning models in managing structured, linked data, the field of graph-based machine learning is expanding quickly. Graph-based models provide a new paradigm for predicting tasks as more real-world situations are represented through complex networks (or graphs), where the relationships between elements are crucial. After giving the essential background information on graph-based learning and its evolution, this section reviews relevant research from a variety of fields, including social networks, biological systems, Internet of Things environments, and communication networks.

## 2.1 Network Data Graphs

A graph with $G = (V, E)$

Nodes (or vertices) V and edges E, which show the connections between nodes, make up G=(V,E). Graphs can be used to represent a variety of real-world networks, including:

A graph is a type of data structure where nodes and edges stand in for entities and their relationships, respectively. Since graphs inherently capture the structure and interaction patterns of data, they are extensively useful across a variety of domains. Graphs can show:

- Social networks: where users are nodes and friendships, connections, or interactions are edges.
- Biological networks: In which biological relationships are shown by edges and nodes stand in for molecules, proteins, or genes.
- Communication networks: These networks have nodes, which are devices for communication, and edges, which are data transfers between them.
- Devices (sensors, actuators) make up nodes in Internet of Things networks, whereas connections or interactions between these devices are represented by edges.

Because of their complex, non-Euclidean structure, these networks are challenging for conventional machine learning techniques to manage. Traditional machine learning techniques handle data points separately, disregarding their connections, which makes them inappropriate for a large number of graph-based real-world applications. Graph-based Machine Learning (GML) techniques were developed in response to the requirement to incorporate network structure into machine learning models.

## 2.2 Machine Learning on Graphs

Traditional machine learning models treat data as independent and identically distributed (i.i.d.), but graph data violates this assumption due to the dependencies between nodes. Hence, graph-based machine learning models are designed to work on data where relationships between nodes are critical. The goal is to leverage the graph structure to improve predictions, by learning from both node features and the graph topology. Traditional graph-theoretic algorithms, such as Page Rank, shortest path algorithms, and community recognition techniques, were used in early graph-based predictive modeling systems. Although these algorithms worked well for some network analysis tasks, they were not generalizable and did not provide a means of extracting intricate, high-dimensional patterns from graph data.

By flattening or converting the graph into tabular form, machine learning methods such as Random Forest, Support Vector Machines (SVMs), or logistic regression were applied to graph data, using the attributes of nodes and their immediate neighbors as input. Nevertheless, these changes frequently led to the loss of important structural data that was present in graphs. For example, details about distant neighbors or the graph's global structure are ignored.

**Key techniques in graph-based learning include:**

- Node Classification: Predicting the label or category of nodes in a graph.
- Link Prediction: Predicting whether a link will form between two nodes.
- Graph Classification: Predicting a label or property for an entire graph structure.

## 2.3. GNNs, or graph neural networks

The foundation of contemporary machine learning techniques for graph data is Graph Neural Networks (GNNs). By spreading and aggregating node input from their neighbors, GNNs which were first introduced by Scarselli et al. (2009) extend neural networks to graphs, allowing models to identify both local and global patterns in graph-structured data. Learning node embeddings, which are dense vector representations of nodes that capture the nodes' attributes and location within the graph, is a crucial element of GNNs. Numerous predictive tasks, including node classification, link prediction, and graph classification, can then be performed using these embeddings. Like conventional neural networks, GNNs are trained by back propagation; however, they use a unique message-passing method to spread information throughout the graph.

### 2.4. GCNs, or graph convolutional networks

One of the most often used extensions of GNNs is Graph Convolutional Networks (GCNs), which were first presented by Kipf and Welling (2017). By establishing a convolutional filter that gathers data from a node's local vicinity, GCNs adapt convolution operations to graph structures, drawing inspiration from convolutional neural networks (CNNs), which are excellent at processing grid-structured data like photographs.

For problems like semi-supervised learning on graphs, where the objective is to predict the labels of the remaining nodes after some of the nodes have been tagged, GCNs have been widely employed. Graph convolution procedures and effective propagation rules enable GCNs to scale to enormous graphs, which is one of its main advantages.

**Important Uses for GCNs:**

1. Social networks: By anticipating new connections between users, GCNs are utilized in friend recommendation systems.
2. Biological networks: By simulating intricate molecular connections, GCNs are used to predict protein function and find new drugs.
3. Communication networks: By examining traffic patterns among network devices, GCNs enhance anomaly identification.

### 2.5. GATs, or graph attention networks

By introducing the attention mechanism to graph-based learning, Graph Attention Networks (GATs), as proposed by Velickovic et al. (2018), enable models to concentrate on the most significant neighbors when aggregating data. GATs learn various weights for different neighbors according on their relevance, in contrast to GCNs that apply the same weighting to all neighbors. It has been demonstrated that this attention mechanism enhances performance on a number of tasks, particularly when neighbors' relative relevance fluctuates among nodes. GATs are especially helpful for forecasting such complex relationships since, for instance, in social networks, a user's influence may rely more on a small number of close friends than on the larger social circle.

### 2.6. Auto encoders for graphs (GAEs)

With Graph Auto encoders (GAEs), an unsupervised learning approach, the model attempts to reconstruct the graph from latent node embeddings. GAEs employ an encoder-decoder architecture in which the decoder attempts to rebuild the graph from low-dimensional node representations that the encoder has learned. For tasks like link prediction and network data anomaly detection, GAEs have shown themselves to be successful. For example, they are commonly employed in communication networks to identify unusual traffic flows and in biological networks to forecast missing links in molecular interaction networks.

### 2.7. Related Studies in Different Fields

- 2.7.1. Social Media One of the most well-known uses of graph-based machine learning has been predictive modeling in social networks. GNNs and GCNs have been used by researchers to categorize content, anticipate user behavior, and pinpoint influential users. By taking use of the social structure of these networks, graph-based models have greatly enhanced tasks like fraud detection, community discovery, and recommendation systems.
- 2.7.2. Networks of Biological Systems Networks in biological systems are essential for comprehending metabolic processes, disease causes, and gene-protein interactions. Applications of GNNs and GAEs include gene classification, drug discovery, and the prediction of protein-protein interactions. Zitnik et al. (2018) greatly increased the accuracy of illness predictions by predicting disease-gene connections using graph-based algorithms.
- 2.7.3. Internet of Things Networks Large-scale network data is produced by the growing number of IoT devices, and machine learning models must take dynamic device interactions into consideration. In the

Internet of Things, graph-based models are employed for predictive maintenance, anomaly detection, and defect identification. Gao et al. (2020), for instance, used GCNs to simulate the interaction patterns of connected devices in order to identify malicious activities in smart home networks.

- 2.7.4. Networks for Communication In communication systems, network security, defect detection, and traffic prediction have all been revolutionized by graph-based predictive modeling. GNNs can detect anomalous patterns that indicate network malfunctions or security breaches by modeling communication equipment as nodes and their interactions as edges. Diro et al. (2018) used network traffic patterns to show how GNNs can be used to detect distributed denial of service (DDoS) attacks.

## 3. GNNS, OR GRAPH NEURAL NETWORKS

Deep learning architectures called Graph Neural Networks (GNNs) are made to function directly with graph structures. By adding the idea of message transfer between nodes, GNNs expand on conventional neural networks. Every node collects data from its neighbors and modifies its representation in light of the data they provide. A major advancement in machine learning, graph neural networks (GNNs) are made especially to process graph-structured data. The intricate interactions and relationships between data points in graphs are difficult for traditional machine learning models to grasp because they treat data as independent and identically distributed (i.i.d.). GNNs, on the other hand, are made especially to operate with graph data by utilizing the graph's structure as well as the node properties. Because of this, GNNs are very good at a number of predictive tasks, such as graph classification, node classification, and link prediction.

A GNN adheres to the propagation rule mathematically:

$$h_v^{(k)} = \text{Agg}\left(h_u^{(k-1)} : u \in N(v)\right)$$

Where $h_v^{(k)}$ is the embedding of node $v$ at layer $k$, $N(v)$ represents the neighbors of $v$, and Agg is an aggregation function (e.g., sum, mean, max).

GNNs can be applied to a variety of tasks, including **node classification** (labeling nodes), **graph classification** (labeling entire graphs), and **link prediction** (predicting relationships between nodes).

### 3.1.1 GNNs: Concept and Architecture

Fundamentally, GNNs use a technique called message passing to aggregate data from nearby nodes in order to learn node embeddings, which are dense vector representations of every node in a network. By merging its own feature representation with that of its nearby nodes, every node in a GNN modifies its embedding.

The following is a mathematical description of a GNN. Let a network with nodes V and edges E be represented by $G = (V, E)$ G=(V,E), and let $hv\ (l)$ h v (l) denote the embedding of node v at layer l. A GNN uses a recurrent process of transformation and aggregation to update the node embeddings:

1. **Message Passing/Aggregation:** Each node aggregates the embeddings of its neighbors to capture information from its local neighborhood. Formally, the aggregated message for node $v$ at layer $l$ is:

$$m_v^{(l)} = \text{Agg}\left(h_u^{(l)} : u \in N(v)\right)$$

Where $N(v)$ is the set of neighbors of node $v$, and Agg is an aggregation function such as mean, sum, or max.

2. **Node Update/Transformation:** After aggregation, each node updates its embedding based on the aggregated information from its neighbors and its own current embedding. The update rule is typically a function such as:

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{Concat}(h_v^{(l)}, m_v^{(l)})\right)$$

Where $W^{(l)}$ is a learnable weight matrix, $\sigma$ is a non-linear activation function (e.g., ReLU), and Concat denotes concatenation of the node's own features with the aggregated neighbor features.

Following figure 1 represents conceptual diagram of the architecture of a Graph Neural Network (GNN), illustrating how nodes and edges interact in a graph structure, with message-passing and feature aggregation layers.

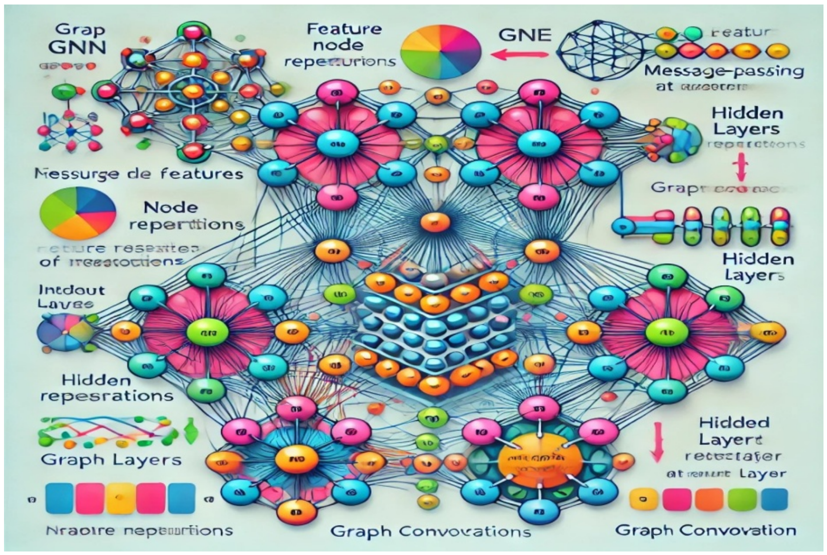


Fig 1: conceptual diagram of the architecture of a Graph Neural Network (GNN), illustrating how nodes and edges interact in a graph structure, with message-passing and feature aggregation layers.

After that, the final node embeddings can be applied to a number of downstream activities, including:

1. Node classification: Ascertaining whether a user in a social network is a member of a particular group by predicting a label for each node.
2. Predicting the possibility that an edge will form between two nodes (for example, friend recommendations in a social network) is known as link prediction.
3. Predicting a label for the complete graph is known as graph classification (e.g., determining whether a chemical graph represents a dangerous molecule).

### 3.1.2 GNN Model Types

There are various GNN variants, each intended to handle particular jobs or enhance the effectiveness and precision of embedding learning and message passing:

#### (a) GCNs, or graph convolutional networks

The idea of conventional convolutional neural networks (CNNs) is extended to graph structures by graph convolutional networks (GCNs). GCNs carry out convolutions across a node's local neighborhood in the graph rather than over a grid-like structure (such as pictures). One of the most well-liked and frequently applied GNN architectures is the GCN model.

According to Kipf and Welling's (2017) proposal, the update rule for GCNs is as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

Where:

- $\tilde{A} = A + I$ is the adjacency matrix with added self-loops (allowing each node to aggregate its own features).
- $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$.
- $H^{(l)}$ is the node feature matrix at layer $l$.
- $W^{(l)}$ is a learnable weight matrix.
- $\sigma$ is an activation function.

GCNs have been effectively used for **semi-supervised learning** on graphs, where only a small subset of nodes are labeled, and the task is to predict the labels of the remaining nodes. This makes them particularly useful for tasks like community detection, content recommendation, and protein function prediction.

#### (b) GATs, or graph attention networks

By adding attention mechanisms to GNNs, Graph Attention Networks (GATs), first presented by Velickovic et al. (2018), enable the model to consider the relative relevance of various neighbors when aggregating data. In contrast to GCNs, which give each neighbor a uniform weight, GATs employ an attention mechanism to give each neighbor a distinct weight, allowing the model to concentrate on the most significant interactions.

The following formula is used to calculate the attention coefficient $\alpha ij$ α ij between nodes i and j:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^T[Wh_i||Wh_j]\right)\right)}{\sum_{k\in N(i)}\exp\left(\text{LeakyReLU}\left(a^T[Wh_i||Wh_k]\right)\right)}$$

Where $a$ is a learnable weight vector, $W$ is a weight matrix, and $||$ denotes concatenation. This mechanism enables GATs to learn the relative importance of neighboring nodes for each node's representation, making them especially useful in cases where certain nodes are more influential than others in predicting a target outcome.

**(c) GraphSAGE**

Hamilton et al. (2017) presented GraphSAGE (Graph Sample and Aggregate), a scalability-focused variation of GNNs. GraphSAGE is more scalable to big graphs since it samples a fixed-size subset of neighbors for each node rather than aggregating data from all neighbors. When managing massive real-world graphs, like social networks with millions of people, this is especially advantageous. LSTM-based aggregation, mean, and pooling-based aggregation are among the various aggregation methods that GraphSAGE can employ. Applications for it include fraud detection in financial transaction networks and link prediction in social networks.

### 3.1.3 GNNs' Benefits

Compared to conventional machine learning models and graph-theoretic techniques, GNNs provide the following benefits:

1. Exploiting Graph Structure: GNNs are able to capture both local and global dependencies among nodes by taking advantage of the rich relational structure of graph data.
2. Generalization: GNNs are suitable for a variety of applications, ranging from node-level predictions to graph-level predictions, due to their ability to generalize across various graph architectures.
3. End-to-End Learning: GNNs can be trained in an end-to-end fashion, which optimizes for the downstream prediction job by learning the node embeddings straight from the graph structure and raw inputs.

### 3.1.4 GNN Applications Due to their adaptability and generalizability, GNNs are widely used in a variety of fields:

1. Social Networks: On social media sites like Facebook and Twitter, GNNs are utilized for impact maximization, community discovery, and friend referral algorithms.
2. Biological Networks: By examining molecular interaction networks, GNNs in bioinformatics aid in the prediction of protein-protein interactions, drug development, and disease gene identification.
3. Communication Networks: GNNs are used in communication systems, particularly when handling data from Internet of Things devices, for defect detection, anomaly detection, and network traffic prediction.
4. Recommendation Systems: By identifying intricate patterns of user-item interaction in content platforms and e-commerce, GNNs improve recommendation algorithms.

### 3.1.5 Restrictions and Difficulties

Despite their great effectiveness, GNNs have several drawbacks.

1. Scalability: GNNs frequently have trouble scaling, especially when dealing with graphs that include millions or billions of nodes. Although they can lessen this, sampling strategies like the ones employed in GraphSAGE may result in information loss.
2. Over-smoothing: Node representations in GNNs with numerous layers often become indistinguishable (over-smooth), which can reduce prediction accuracy.
3. Dynamic Graphs: While most GNNs assume static graphs, many real-world graphs are dynamic, meaning they change over time. It is still a major research issue to extend GNNs to dynamic situations.

## 3.2 Convolutional networks based on graphs (GCNs)

Conventional convolutional neural networks are extended to graph structures by Graph Convolutional Networks (GCNs). GCNs apply convolutional filters across the local neighborhoods of the graph rather than convolutions to grid-like structures (like pictures).

The GCN propagation rule can be expressed as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

Where:

- $\tilde{A}$ is the adjacency matrix with added self-loops.
- $\tilde{D}$ is the diagonal node degree matrix.
- $H^{(l)}$ is the node feature matrix at layer $l$.
- $W^{(l)}$ is the layer's learnable weight matrix.
- $\sigma$ is a non-linear activation function.

GCNs are particularly powerful for tasks such as **semi-supervised node classification** and

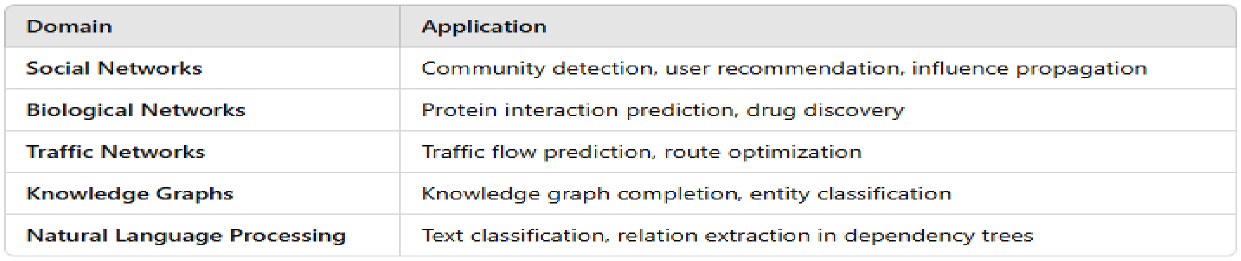
Table 1 shows Key Components of Graph Convolutional Networks (GCNs)

| Component | Description |
|---|---|
| **Node Features** | Initial feature vectors assigned to each node (e.g., sensor readings, text data). |
| **Graph Structure** | Defined by the adjacency matrix representing the connections (edges) between nodes. |
| **Convolutional Layers** | Aggregation of neighbor node features using graph convolution operations. |
| **Weight Matrix (W)** | Learnable parameters that transform node features at each layer. |
| **Activation Function** | Non-linear function (e.g., ReLU) applied after aggregation for feature transformation. |
| **Normalization** | Ensures that the influence of each node's neighbors is normalized, often using the degree matrix. |
| **Output Layer** | Produces the final output, such as node classification or link prediction. |
| **Loss Function** | Measures the error between predictions and true labels, used for model optimization. |

Table 1: Key Components of Graph Convolutional Networks (GCNs)

| Feature | Traditional CNNs | Graph Convolutional Networks (GCNs) |
|---|---|---|
| **Input Structure** | Grid-like, usually 2D (e.g., images) | Graph structure with nodes and edges |
| **Neighborhood Definition** | Local pixel neighborhoods (e.g., 3x3 filter) | Defined by graph connectivity (adjacent nodes) |
| **Convolution Operation** | Slide filter across grid | Aggregate features from neighbors via weighted sum or mean |
| **Applications** | Image recognition, video analysis, etc. | Social networks, recommendation systems, biological networks |
| **Data Type** | Euclidean (structured data) | Non-Euclidean (structured and unstructured data) |
| **Scalability** | Efficient with large, regular grids | Challenging with large, irregular graphs |

Table 2: Comparison of Traditional CNNs and Graph Convolutional Networks (GCNs)



| Domain | Application |
|---|---|
| Social Networks | Community detection, user recommendation, influence propagation |
| Biological Networks | Protein interaction prediction, drug discovery |
| Traffic Networks | Traffic flow prediction, route optimization |
| Knowledge Graphs | Knowledge graph completion, entity classification |
| Natural Language Processing | Text classification, relation extraction in dependency trees |

Table 3: Key Applications of GCNs

By successfully applying the ideas of conventional convolutional neural networks (CNNs) to graph-structured data, graph convolutional networks (GCNs) constitute a significant breakthrough in graph-based machine learning. By extending the convolution operation to any graph structures, GCNs make it possible to learn on non-Euclidean data, such graphs. Since its introduction by Kipf and Welling (2017), GCNs have become increasingly popular because of their scalable and effective use of the graph's structure as well as node properties.

### 3.2.1 Graph Convolutions' Essential Ideas

Regular grid-like data structures, such 2D pictures, where each pixel has a defined number of neighbors, are subjected to convolutions in ordinary CNNs. On the other hand, nodes in graphs may have varying numbers of neighbors, and the structure of the links between nodes varies. By combining data from a node's immediate neighborhood and updating its representation using this combined data, GCNs reinterpret the convolution operation to operate on graphs in order to overcome these difficulties.

The following is a GCN's basic function:

1. Neighbor Information Aggregation: The characteristics of nearby nodes are combined for every node. By doing this, the node is able to extract contextual data from its local graph structure.
2. Feature Transformation: The new representation of the node is calculated by applying a transformation to the aggregated data after aggregation, usually a linear transformation followed by a non-linearity.

**The layer-wise propagation rule of a GCN can be expressed mathematically as follows:**

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

Where:

- $H^{(l)}$ is the matrix of node features at layer $l$.
- $\tilde{A} = A + I$ is the adjacency matrix with added self-loops, where $A$ is the original adjacency matrix of the graph, and $I$ is the identity matrix (adding self-loops allows each node to incorporate its own features during the update).
- $\tilde{D}$ is the diagonal degree matrix corresponding to $\tilde{A}$, where each element $\tilde{D}_{ii}$ is the sum of the degrees of node $i$ plus one (accounting for self-loops).
- $W^{(l)}$ is the learnable weight matrix for layer $l$.
- $\sigma$ is a non-linear activation function, such as ReLU.

Following figure 2 shows the diagram illustrating the basic function of Graph Convolutional Networks (GCNs), showcasing how node features are aggregated through graph convolution layers.

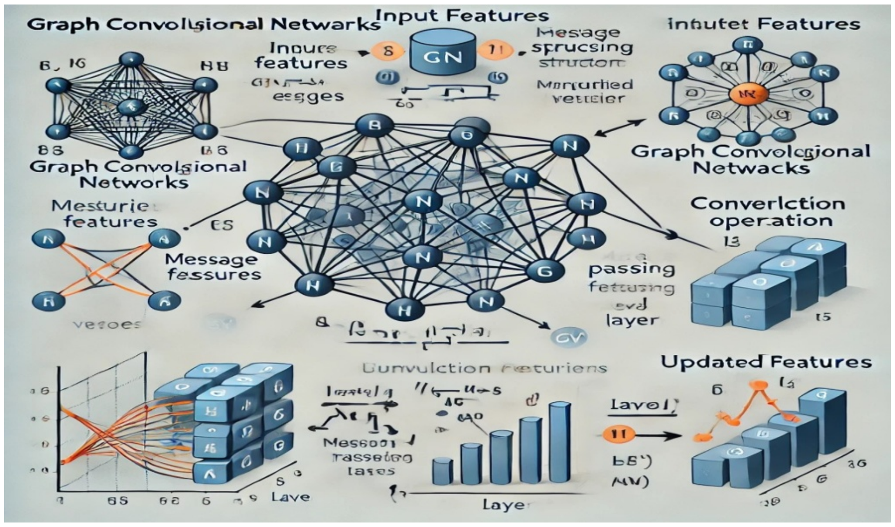


Fig 2: GCN's basic function

### 3.2.2 GCNs' Characteristics and Advantages

Traditional convolutional neural networks give GCNs a number of significant characteristics that make them ideal for graph data:

1. Locality: By combining data from each node's immediate neighborhood, GCNs are able to identify local patterns in graphs, much like CNNs' convolutional filters do with images.
2. Weight Sharing: GCNs share weights among several nodes and their surrounding areas, much like CNNs do when applying the same filter to various areas of an image. GCNs are more parameter-efficient thanks to this weight-sharing method, particularly when used on big networks.
3. Learning Hierarchical Features: The model may represent both local and global graph structures by combining data from ever bigger neighborhoods through the stacking of numerous GCN layers.

## 3.3 GATs, or graph attention networks

By adding attention mechanisms to GNNs, Graph Attention Networks (GATs) enable the model to give various neighbors varying weights according to how important they are to the target node. The attention coefficient between nodes I and J in GATs is calculated as follows:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^T[Wh_i||Wh_j]\right)\right)}{\sum_{k\in N(i)} \exp\left(\text{LeakyReLU}\left(a^T[Wh_i||Wh_k]\right)\right)}$$

This allows GATs to weigh neighbors differently based on learned patterns in the data, making them useful for tasks where certain relationships are more significant than others.

## 3.4 Auto encoders for graphs (GAEs)

Unsupervised learning models called Graph Auto encoders (GAEs) are used to learn graph latent representations. They employ a decoder to rebuild the graph from the latent embeddings after mapping nodes into a latent space using an encoder. GAEs are frequently employed in anomaly detection and link prediction. Graph Auto encoders (GAEs) are a subclass of Auto encoders, a class of neural networks mostly used for unsupervised learning that have been modified for graph data. GAEs preserve crucial structural and feature-related information while learning low-dimensional, compact representations (embeddings) of network nodes. Following that, these embeddings can be applied to a number of downstream tasks, including graph construction, link prediction, and graph clustering. GAEs are a potent tool for learning from intricate and linked systems because they expand the traditional Auto encoder's framework to operate with graph-structured data.

An encoder and a decoder are the two primary parts of Auto encoders.

1. Encoder: The encoder efficiently compresses the data while maintaining important characteristics by mapping the input data in this case, graph nodes and their features—to a lower-dimensional latent space.
2. Decoder: To make sure that the learnt embeddings contain the most important details about the graph structure and node attributes, the decoder tries to recover the original data from the latent representation.

Reconstruction error, or the discrepancy between the original and recreated data, is what an auto encoder aims to decrease.

### 3.4.2 Auto encoders for graphs (GAEs)

A graph the concept of Auto encoders is extended to graph data by Auto encoders. Similar to GCNs, the encoder in a GAE learns node embeddings by combining data from the local graph neighborhood. The decoder then uses these embeddings to reconstruct the graph structure or other graph-related attributes.

The following is a description of GAEs' general workflow:

1. Encoding: By using a graph-based aggregation function (such as GCN layers), the GAE encoder learns node embeddings using the graph structure (adjacency matrix) and node attributes as input. The graph data is compressed into a lower-dimensional space in this step.
2. Decoding: The decoder aims to anticipate whether edges (links) exist between nodes by reconstructing the graph's adjacency matrix using the learned embeddings. For assignments such as link prediction, the decoder outputs the likelihood of edges between nodes, based on their embeddings.

Formally, let $G = (V, E)$ represent a graph with nodes $V$ and edges $E$. The GAE model is structured as follows:

- **Encoder:** The encoder is typically based on a Graph Convolutional Network (GCN), which computes node embeddings $Z \in \mathbb{R}^{|V| \times d}$, where $d$ is the dimension of the embedding space. The node embeddings $Z$ are learned by aggregating features from neighboring nodes, as done in GCNs.

$$Z = \mathrm{GCN}(A, X)$$

Where:

- $A$ is the adjacency matrix of the graph.
- $X$ is the feature matrix.
- **Decoder**: The decoder reconstructs the graph by predicting the adjacency matrix $\hat{A}$ from the node embeddings. The reconstruction can be done by computing the dot product between node embeddings, which represents the likelihood of an edge between two nodes:

$$\hat{A}_{ij} = \sigma(Z_i^T Z_j)$$

Where $Z_i$ and $Z_j$ are the embeddings of nodes $i$ and $j$, and $\sigma$ is a sigmoid function that outputs a value between 0 and 1 (representing the probability of an edge between $i$ and $j$).

The GAE is trained to minimize the reconstruction loss, which measures the difference between the predicted adjacency matrix $\hat{A}$ and the actual adjacency matrix $A$.

### 3.4.3 Auto encoders for Variational Graphs (VGAEs)

The Variational Graph Auto encoders (VGAE), first presented by Kipf and Welling (2016), is a noteworthy extension of GAEs. By adding probabilistic modeling, VGAEs expand on the conventional GAE and enable more adaptable and reliable embeddings.  Instead of producing deterministic embeddings, the encoder in a VGAE creates a distribution over the latent embeddings. The objective is to sample embeddings from this distribution and use the sampled embeddings to reconstruct the graph. VGAEs are especially helpful for jobs requiring uncertainty estimates, such graph construction and anomaly detection, because of their probabilistic architecture.

The key components of VGAEs are:

- **Probabilistic Encoder**: Instead of learning deterministic embeddings, the encoder learns the parameters (mean $\mu$ and variance $\sigma^2$) of a Gaussian distribution over the embeddings for each node:

$$q(Z|X,A) = \prod_{i=1}^{|V|} \mathcal{N}(Z_i|\mu_i, \sigma_i^2)$$

- **Sampling**: Embeddings are sampled from this distribution, typically using the **reparameterization trick** to enable backpropagation.
- **Decoder**: The decoder remains similar to the GAE, reconstructing the adjacency matrix from the sampled embeddings.

The variational lower bound is used to train VGAEs, balancing two goals:

1. Reconstruction loss minimization (like GAEs).
2. Reducing the difference between a prior distribution (usually a normal distribution) and the learned distribution, known as the Kullback-Leibler (KL) divergence.

### 3.4.4 GAE applications

Applications of GAEs and VGAEs to a variety of applications using graph-structured data have proven fruitful. Among the important uses are:

#### (a) Predicting Links

1. One of the most frequent jobs for GAEs is link prediction. Predicting whether an edge already exists between two nodes or whether a new edge will form in the future is the aim of this work. Because GAEs directly train latent representations that capture the likelihood of connections between nodes, they are especially well-suited for this job.
2. A GAE, for instance, can be used to suggest possible friendships in a social network by taking into account the network's structure and the users' learnt embeddings. GAEs can also forecast how genes or proteins will interact in biological networks.

#### (b) Finding Anomalies

By locating nodes or edges in a network that substantially depart from typical patterns, GAEs can be used for anomaly identification. Analyzing the graph's reconstruction inaccuracy will help achieve this. High reconstruction error nodes or edges are probably abnormalities because the GAE finds it difficult to simulate their activity in the graph. This is especially helpful for applications like financial transaction networks' fraud detection, where unusual transactions or organizations can be identified based on their deviations from the expected network structure.

### (c) Generating Graphs

In graph generation jobs, where the objective is to create new graphs with structural characteristics similar to a given collection of graphs, GAEs can also be modified. This is helpful in situations like drug discovery, where creating molecular graphs with the correct chemical characteristics is the aim.

### (d) Clustering of Graphs

The objective of graph clustering is to organize nodes into clusters according to structural and feature similarity. Effective node clustering is made possible by GAEs' ability to learn embeddings that capture both local and global graph structure. This can be used in a number of fields, such as topic modeling in citation networks and community detection in social networks.

### 3.4.5 GAEs' Advantages and Drawbacks

Strengths Unsupervised Learning: Since GAEs don't need labeled data, they're perfect for unsupervised tasks like graph clustering and link prediction.

1. Dimensionality Reduction: For tasks like large-scale graph analysis that need for effective computing and storage, GAEs offer a compact representation of nodes.
2. Flexibility: GAEs are very adaptable for a variety of applications since they may be used with undirected, directed, weighted, and attributed graphs, among other graph types.

**Restrictions**

1. Scalability: Because the encoder must compile data from every neighbor for every node, GAEs, like other GNN-based models, may have trouble scaling when used on big graphs.
2. Over fitting: When the training graph is tiny or the embeddings are overly expressive, GAEs, especially in their vanilla version, are susceptible to over fitting.
3. Focus on Reconstruction: GAEs are made to reduce reconstruction loss, which might not always be in line with what some downstream activities are trying to achieve. Therefore, when applying GAEs to particular applications, fine-tuning can be required.

## 4. GRAPH-BASED MACHINE LEARNING APPLICATIONS

Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), and Graph Auto encoders (GAEs) are examples of graph-based machine learning models that have demonstrated exceptional effectiveness in resolving challenging tasks in a variety of disciplines. In order to acquire meaningful node and graph representations and perform well on tasks involving relational data, these models take advantage of the structural aspects of graphs. This section will examine important graph-based machine learning applications across various domains, demonstrating their adaptability and efficiency.

### 4.1 Social Media

In social networks, graph-based machine learning algorithms are widely employed for applications such as recommendation systems, influence prediction, and community recognition. In applications like content suggestion and friendship suggestions, GNNs and GCNs enhance the performance of predictive models by assisting in the capture of complex user relationships.

### 4.1.1 Identifying Communities

Finding clusters of nodes in a network that are more closely related to one another than to the rest of the graph is known as community discovery. Communities in social networks, such online friend groups or professional networks are frequently groups of people who have similar interests. Communities can be found using GNNs, particularly Graph Convolutional Networks (GCNs), which learn node embeddings that represent the graph structure and node properties. Graph-based algorithms, for instance, are able to recognize user groups on social media sites like Facebook and Twitter by using networks of friendships or interactions. This has important ramifications for influence analysis, recommendation engines, and targeted information distribution.

### 4.1.2 Predicting Links

The goal of link prediction is to forecast if there are any missing links in a network or whether new connections will likely arise between nodes. This is very helpful in social networks when suggesting new acquaintances or connections. Because they can estimate the probability of linkages between nodes by learning compact node embeddings, graph Auto encoders (GAEs) are ideally suited for this purpose. By examining a user's current network and identifying possible connections based on shared connections and common attributes, LinkedIn, for example, use link prediction techniques to recommend new professional connections.

## 4.2 Networks of Biological Systems

In biological networks, connections are represented by edges, and biological things such as genes or proteins are represented by nodes. Graph-based models are employed in illness prediction (categorizing genes according to disease connections), drug development (predicting drug-target interactions), and protein function prediction. In biological networks, GAEs have shown promise in anticipating missing connections.

### 4.2.1 Predicting Protein-Protein Interactions (PPI)

Protein-protein interaction (PPI) networks and other biological networks can be represented as graphs with nodes standing in for proteins and edges for interactions between them. Understanding biological processes and identifying novel therapeutic targets depend on the ability to predict novel protein interactions. By learning from the graph structure and using the learnt embeddings to predict novel protein interactions, graph convolutional networks (GCNs) and variational graph Auto encoders (VGAEs) have demonstrated significant potential in this field. GCNs, for instance, can be used in bioinformatics to analyze vast networks of protein-protein interactions and uncover hitherto unidentified interactions, advancing precision medicine and drug development.

### 4.2.2 Association of Genes with Disease

Predicting disease-gene relationships is a significant use of graph-based learning in biological networks. A bipartite graph that shows known relationships between genes and diseases can be used to illustrate genes and diseases. New gene-disease connections can be predicted thanks to GNNs' ability to learn embeddings for both genes and disorders. In order to determine the genetic causes of diseases, experimental study can be guided by these predictions. GNNs, for example, are used in genomics to predict which genes are linked to illnesses like cancer, offering information that might guide the development of novel treatment approaches.

## 4.3 Networks for Communication

Graph-based models are used to forecast traffic flows, identify abnormalities (such as faults or intrusions), and optimize routing in communication networks. Because GNNs comprehend the intricate relationships in communication infrastructures, they enable models to forecast network outages or security breaches.

### 4.3.1 Predicting Molecular Properties

Predicting a molecule's characteristics, such as its solubility, toxicity, and binding affinity, is essential in drug discovery in order to find prospective medication candidates. Atoms are nodes and chemical bonds are edges in a graph that represents molecules. For learning molecular embeddings that represent the chemical structure and properties of molecules, graph neural networks (GNNs), in particular graph convolutional networks (GCNs) and message passing neural networks (MPNNs), are quite successful. For instance, GNNs have been used in pharmaceutical research to assess a molecule's potential as a therapeutic ingredient or to forecast if it would bind to a particular protein target. By reducing the number of compounds that require experimental testing, this shortens the time and expense associated with drug discovery.

### 4.3.2 Predicting Drug-Drug Interactions

Given that some medicine combinations can result in negative side effects, anticipating possible drug interactions is crucial to maintaining patient safety. Drug interaction networks, in which medications are represented by nodes and known interactions by edges, can be modeled using GNNs. Based on the network structure and the characteristics of the medications, GNNs can forecast novel interactions by learning node embeddings. GNN-based models, for example, aid in the prediction of unfavorable medication interactions in the healthcare industry, enabling safer and more individualized treatment regimens.

### 4.4 The Internet of Things

Graph-based models can be used to identify abnormalities and forecast failures in the complex network data generated by the Internet of Things ecosystem, which is made up of interconnected devices. In the Internet of Things, graph-based anomaly detection is especially useful for identifying odd activity, including security breaches or device issues.

A network of linked devices that gather, share, and process data is known as the Internet of Things (IoT). Sensors, appliances, automobiles, and several other systems with embedded electronics, software, and network connectivity are examples of these devices. Graph-based machine learning techniques are ideally suited for IoT networks due to their dynamic nature and intricate relationships. We may apply Graph Neural Networks (GNNs) and other graph-based models to a range of IoT-related tasks by representing IoT systems as graphs, where devices and their relationships constitute nodes and edges.

#### 4.4.1 Communication Networks and Device Interaction

Devices communicate with one another in IoT networks by exchanging data or initiating activities. Each device is a node in a graph that depicts these interactions, and the edges show instances of data sharing or communication between the devices. By modeling these interactions, GNNs can be used to improve IoT network administration, prediction, and monitoring.

**(a) Identification of Anomalies in IoT Networks**

Because IoT networks are vast and dispersed, it is essential to identify anomalies like broken devices, strange traffic patterns, or possible security breaches. Device communication patterns can be examined and anomalies from typical behavior can be found using graph-based models. Graph Convolutional Networks (GCNs) or Graph Auto encoders (GAEs) can detect suspicious activity, like unexpected device interactions or anomalous data flows, by learning the network's regular topology. GNNs, for instance, can identify communication irregularities among smart devices in smart home systems, highlighting security risks like illegal network access or possible appliance faults.

**(a) Analysis of Network Traffic**

IoT devices create intricate communication patterns by continuously generating and exchanging data. It is possible to monitor traffic and anticipate any bottlenecks or network failures by using graph-based models, which depict this data flow as a graph with nodes representing devices and edges indicating data exchanges. For real-time traffic analysis, Graph Attention Networks (GATs) are very helpful since they may concentrate on the most important connections in the network. Graph-based models can be used in smart cities to predict and manage traffic congestion or optimize resource allocation by analyzing the data flow between IoT-enabled infrastructure, including environmental sensors, traffic signals, and surveillance cameras.

#### 4.4.2 Predictive Maintenance IoT devices

Is frequently a component of big, intricate systems, such as smart buildings or manufacturing facilities. To avoid device failures, these systems need to be continuously monitored. The process of forecasting a device's likelihood of failure using the historical and current data it produces is known as predictive maintenance. Device relationships can be modeled using graph-based models, which can also be used to identify patterns that point to possible failure or wear and tear. GNNs are able to learn the underlying dependencies and interactions in the system by using a graph representation of devices and their operational states. By spotting irregularities in the patterns of device interaction, this enables more precise failure predictions. Sensors connected to industrial machinery in a smart factory, for example, can be represented as a graph in which each machine's state is impacted by its interactions with nearby machines. GNNs can forecast a machine's likelihood of failure, allowing for prompt maintenance and less downtime.

### 4.4.3 Data Fusion and Sensor Networks

IoT networks frequently have a lot of sensors that track various environmental parameters including motion, temperature, humidity, and air quality. Graphs can be used to depict sensor networks, with sensors serving as nodes and edges signifying data correlation or communication between them. By combining data from several sensors, graph-based models can increase forecast accuracy and dependability.

#### (a) Monitoring the Environment

Sensor networks gather information on a variety of environmental characteristics over a large geographic area in environmental monitoring systems. GNNs are able to discover spatial relationships between various sensor values by representing the sensor network as a graph. This allows for more precise environmental forecasts and anomaly detection, including spotting regions with unusually high pollution levels or forecasting weather patterns using sensor data. IoT-enabled sensors that track temperature, humidity, and soil moisture in agriculture, for instance, can be shown as a graph. By analyzing this sensor data, GNNs can optimize irrigation schedules, increasing agricultural production while using less water.

#### (b) Intelligent Medical Systems

IoT devices in the healthcare industry, such wearable sensors and medical monitoring equipment, produce a lot of data about a patient's condition. Graph-based models can facilitate data fusion from many sensors, offering a more thorough picture of a patient's health, by representing these devices as a graph, where nodes are devices and edges reflect the flow of health-related data. For example, GNNs can combine information from wearable devices that measure blood pressure, heart rate, and activity levels in remote patient monitoring systems, allowing for the early identification of health abnormalities like hypertension or arrhythmias.

## 5. DIFFICULTIES AND PROSPECTS FOR FURTHER RESEARCH

### 5.1.1 Computational Complexity and Scalability

Scalability is one of the main obstacles to using graph-based machine learning in large-scale networks. Millions or even billions of nodes and edges make up many real-world graphs, as those in social networks, financial networks, and Internet of Things systems. Iterative message passing between nodes is usually necessary for graph-based models like GNNs, and as the graph gets bigger, this becomes computationally costly. The capacity of existing models to effectively manage huge networks is limited by the storage and processing requirements of such large-scale graphs, which can overwhelm conventional computing capabilities.

1. Problems: Memory and storage constraints: Node features, edge relationships, and intermediate calculations take up a lot of memory as graphs get bigger.
2. Time complexity: Iterative procedures like message-passing, which are common in graph-based models, have a large computational overhead and scale poorly with graph size.
3. Prospects for Future Research: Techniques for partitioning and graph sampling: Future studies can concentrate on enhancing methods for graph sampling that minimize the size of the input graph while maintaining its structural characteristics. Another potential field of study is graph partitioning techniques, which split big graphs into smaller sub graphs for processing in parallel.
4. GNN topologies that are scalable: Scalability problems can be lessened by creating scalable architectures that work on sub graphs or employ neighborhood sampling, as GraphSAGE and Cluster-GCN.
5. Distributed computing: Another way to get over scalability issues is to investigate the usage of distributed computing frameworks such as Apache Spark or DGL (Deep Graph Library) for effective graph processing.

### 5.1.2 Heterogeneity in Graphs in the Real World

Heterogeneity is a common aspect of real-world graphs, where nodes and edges have varying kinds, attributes, and connections. Sensors, gadgets, and communication channels, for instance, may have unique characteristics in Internet of Things networks. Genes and proteins interact in a variety of intricate ways inside biological networks. Because they presume consistent node types and edge interactions, traditional graph-based models frequently find it difficult to handle such variability.

**Challenges:**

1. Managing multi-modal data: Graphs in fields like healthcare or the Internet of Things may have nodes and edges with various properties, necessitating models that can integrate many data kinds.
2. Heterogeneous graph representation: Since standard GNNs are made for homogeneous networks, it might be challenging to capture the rich semantics of heterogeneous graphs.

**Opportunities for Additional Study:**

1. HGNNs or heterogeneous GNNs: Creating models especially for diverse graphs is one exciting avenue for future research. Heterogeneous Graph Attention Networks (HANs) are one example of a model that may learn various node embeddings and edge interactions.
2. Multi-modal graph learning: To provide a more comprehensive depiction of heterogeneous systems, future studies can examine the efficient integration of various data modalities (such as text, pictures, and sensor data) into graph learning models.

## 5.2 Opportunities for Additional Study

There are a number of fascinating prospects for developing the subject of graph-based machine learning in addition to tackling the aforementioned difficulties. These opportunities present significant promise for further study and real-world implementations:

### 5.2.1 Graph-Based Education for Privacy-Preserving Federated Systems

1. Federated learning approaches that enable models to be trained on decentralized data without exchanging sensitive information are becoming more and more necessary as data privacy concerns develop, especially in industries like healthcare and finance. Federated learning environments, where nodes in the graph represent various entities or data sources that are unable to directly share their data, could benefit from the adaptation of graph-based models.
2. Research might concentrate on creating methods for privacy-preserving graph learning, in which private information (such as financial transactions or private medical records) is safeguarded throughout the training and inference phases.
3. Federated GNNs: Federated GNNs, in which several devices or organizations work together to train a global graph model while maintaining the privacy of their local data, may also be investigated in future research.

### 5.2.2 Combining Other AI Paradigms with Graph Learning

Future research could focus on integrating graph-based models with other machine learning paradigms including computer vision, natural language processing, and reinforcement learning (RL). This multidisciplinary strategy may open up new graph-based learning applications and capabilities.

1. Graph-based reinforcement learning: In graph-structured contexts like robotics or autonomous driving, combining GNNs with RL can result in more efficient decision-making.
2. Graphs in NLP and vision: By utilizing the structural links between things, graph-based models can be used to enhance tasks such as knowledge graph completion in NLP or scene interpretation in computer vision.

### 5.2.3 Improved Methods for Learning Graph Representation

The effectiveness of graph-based machine learning depends on the caliber of node and graph representations. Learning efficient representations is still difficult, nevertheless, particularly for big, sparse, or densely connected graphs.

1. Self-supervised learning for graphs: Self-supervised learning techniques hold great promise for teaching models meaningful graph representations without the need for labeled data. Research is now being conducted on methods such as contrastive learning for graphs, which maximize the agreement across various graph views.
2. Hyper graphs and higher-order networks: Researching hyper graphs, which have edges that can connect more than two nodes, is another exciting avenue. In order to better represent complex linkages in some areas, such biological systems or multi-agent interactions, research can investigate ways to extend GNNs and related models to hyper graphs and other types of higher-order networks.

## 6. CONCLUSIONS

A strong framework for predictive modeling in complex network data is provided by graph-based machine learning models. These models can greatly increase forecast accuracy in a variety of applications, including as social networks, biological networks, communication systems, and Internet of Things settings, by utilizing the relationships between entities in a network. Addressing issues like interpretability, dynamic graph modeling, and scalability will be crucial to raising the bar in this area as research progresses. In a variety of fields, such as social networks, biological systems, recommendation systems, and the Internet of Things (IoT), graph-based machine learning has proven to be a potent paradigm for predictive modeling in complex network data. Machine learning techniques, including Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), and Graph Auto encoders (GAEs), offer reliable ways to extract significant patterns and insights from intricate, interrelated data, as we have shown throughout this paper.

In order to improve predictive capabilities in tasks like node classification, link prediction, anomaly identification, and community detection, significant developments in graph-based learning have made it possible to model both structural and feature interactions within graphs. These techniques are very relevant for real-time data-driven decision-making, especially for IoT applications where graph-based models have demonstrated significant promise in maximizing network efficiency, detecting abnormalities, and enabling predictive maintenance. Notwithstanding the enormous advancements in the subject, a number of problems still exist, such as those pertaining to dynamic graphs, heterogeneous graph architectures, scalability, and the interpretability of graph-based models. More advancement in fields like dynamic graph neural networks, explainability strategies for more visible model outputs, and graph sampling are necessary to meet these problems. Furthermore, combining graph-based learning with other machine learning paradigms like privacy-preserving federated learning and reinforcement learning opens up fascinating new research directions and practical uses.

References

1. Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. Proceedings of the International Conference on Learning Representations (ICLR).
2. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph Attention Networks. Proceedings of the International Conference on Learning Representations (ICLR).
3. Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. Advances in Neural Information Processing Systems (NeurIPS).
4. Grover, A., & Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
5. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How Powerful are Graph Neural Networks? International Conference on Learning Representations (ICLR).
6. Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: Going beyond Euclidean data. IEEE Signal Processing Magazine, 34(4), 18-42. DOI: 10.1109/MSP.2017.2693418
7. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4-24. DOI: 10.1109/TNNLS.2020.2978386
8. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. Proceedings of the International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/1609.02907

9. Hamilton, W., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in Neural Information Processing Systems (NIPS), 30, 1024-1034. URL: https://arxiv.org/abs/1706.02216
10. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. IEEE Transactions on Neural Networks, 20(1), 61-80. DOI: 10.1109/TNN.2008.2005605
11. Wang, X., Zhang, P., Zhang, J., & Zhang, X. (2019). Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering, 29(12), 2724-2743. DOI: 10.1109/TKDE.2019.2927175
12. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), 974-983. DOI: 10.1145/3219819.3219890
13. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. Proceedings of the International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/1710.10903
14. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. AI Open, 1, 57-81. DOI: 10.1016/j.aiopen.2020.01.001
15. Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., ... & Zaremba, W. (2018). Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261. URL: https://arxiv.org/abs/1806.01261
16. Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/1801.10247
17. Cao, Y., Wang, Y., & He, D. (2019). Auto-graph: Automated graph neural network architecture search. Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM), 1501-1510. DOI: 10.1145/3357384.3358023