MANTECH Publications
Journal of Artificial Intelligence, Machine Learning and Soft Computing
Volume 10, Issue 2, July-December 2025

Graph Neural Networks (GNNs) & Relational Machine Learning (Relational ML)

Tribhuwan S Pathak
Associate Professor
Department of Information Technology
Institute of Computing & Data Sciences
Email: *tribhuvans10pathak@gmail.com*

ABSTRACT

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for processing non-Euclidean data structures, enabling machine learning models to exploit relational information in complex networks. Alongside, relational machine learning (Relational ML) has focused on learning predictive models over structured data with interdependencies between entities. This paper presents a comprehensive review of the foundational concepts, architectures, and recent advancements in GNNs and relational ML. We discuss methodologies such as Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Message Passing Neural Networks (MPNNs), as well as applications in social networks, recommender systems, and bioinformatics. Challenges, limitations, and future directions are also examined, emphasizing scalability, interpretability, and integration with symbolic reasoning. This review aims to provide researchers and practitioners with a consolidated understanding of graph-based learning frameworks.

KEYWORDS: *Graph Neural Networks, Relational Machine Learning, Graph Convolutional Networks, Message Passing, Graph Attention Networks, Knowledge Graphs*

INTRODUCTION

Traditional machine learning models assume data points are independent and identically distributed (i.i.d.), which limits their ability to capture complex relationships in structured data.

Many real-world applications, including social networks, protein interaction networks, recommendation systems, and knowledge graphs, require modeling dependencies among entities.

Graph Neural Networks (GNNs) have emerged as a framework to generalize deep learning to graph-structured data. Unlike conventional neural networks, GNNs can model node features, edge relationships, and global graph structures simultaneously. Relational Machine Learning (Relational ML) extends this paradigm by learning from interdependent data instances, often leveraging probabilistic models or logic-based representations.

This paper reviews the theory, architectures, and applications of GNNs and relational ML, highlighting both the potential and limitations of these approaches. Figure 1 illustrates the difference between traditional neural networks and graph-based models.

2. BACKGROUND

2.1 Graph Theory Fundamentals

Graphs are mathematical structures used to represent relationships between entities. Formally, a **graph** is defined as G=(V,E)G = (V, E)G=(V,E), where:

- VVV is the set of **nodes (vertices)**, representing entities such as users, proteins, or documents.
- EEE is the set of **edges**, representing the relationships or interactions between these entities.

Graphs can vary in structure and properties, depending on the nature of the problem domain:

1. **Directed vs. Undirected Graphs:**
   - **Directed graphs** have edges with a specific direction, indicating asymmetric relationships (e.g., Twitter follow networks).
   - **Undirected graphs** treat edges as bidirectional, modeling mutual relationships (e.g., Facebook friendships).
2. **Weighted vs. Unweighted Graphs:**
   - **Weighted graphs** assign a numerical value to edges, representing strength, cost, or similarity (e.g., communication frequency).
   - **Unweighted graphs** treat all edges equally, indicating merely the presence or absence of a relationship.

3. **Attributed Graphs:**
   - Nodes and edges may have **features or attributes** (e.g., age, location for users, or bond type in molecules). These features are often used as input for graph-based learning models.

Common types of graphs in real-world applications include:

- **Social networks:** Users are represented as nodes, and edges indicate social interactions such as friendships, likes, or messages. Graph analysis can detect communities, influential users, or patterns of information diffusion.
- **Knowledge graphs:** Nodes represent entities (e.g., countries, people, organizations) and edges represent semantic relationships (e.g., "born_in," "works_at"). Knowledge graphs are widely used in search engines, recommendation systems, and question-answering systems.
- **Molecular graphs:** Nodes represent atoms, edges represent chemical bonds. Molecular graphs are critical for drug discovery, property prediction, and understanding chemical reactions.

Graph-based learning challenges: Learning on graphs is inherently more complex than traditional tabular data due to:

- **Neighborhood structures:** Nodes are connected in intricate patterns; learning requires aggregating information from neighbors.
- **Connectivity patterns:** Global structure, such as community clusters or central hubs, affects the flow of information and the influence of nodes.
- **Dynamic changes:** Some graphs, such as social networks, evolve over time, requiring models to adapt to changing topologies.

Graph theory provides the foundation for many algorithms in graph analytics, including shortest-path algorithms, centrality measures, spectral graph theory, and community detection, all of which are fundamental for understanding and designing graph neural networks.

2.2 Relational Machine Learning

While graph theory provides a formal structure to represent entities and relationships, **Relational Machine Learning (Relational ML)** focuses on learning predictive models that

exploit these structures. Relational ML models are designed to handle **interdependent data**, where the outcome for one entity may depend on related entities.

Core methods in relational ML include:

1. **Probabilistic Relational Models (PRMs):**
   - Extend Bayesian networks to relational domains, enabling probabilistic reasoning over entities and their relationships.
   - Example: Predicting whether a user will like a product based on their friends' preferences.
2. **Markov Logic Networks (MLNs):**
   - Combine probabilistic graphical models with **first-order logic**, allowing the integration of relational knowledge and uncertainty.
   - Example: Encoding rules such as "Friends of friends are likely to be friends" with associated probabilities.
3. **Factorization-based Models:**
   - Learn low-dimensional **embeddings** of entities and relationships to capture latent features.
   - Examples include **TransE** (models relationships as translations in embedding space) and **DistMult** (models interactions via bilinear forms).
   - Widely used for knowledge graph completion and link prediction.

Applications of relational ML:

- **Link prediction:** Predicting missing relationships, e.g., recommending new friends or identifying potential protein-protein interactions.
- **Node classification:** Assigning labels to nodes based on their features and relationships, e.g., classifying articles in a citation network.
- **Relational clustering:** Grouping nodes based on relational patterns rather than independent features, e.g., discovering communities in social networks.

Relational ML provides the theoretical and practical foundation for modern graph-based learning approaches. By combining structural information from graphs with predictive modeling, it enables advanced AI systems to reason over complex, interconnected datasets.

3. GRAPH NEURAL NETWORKS: ARCHITECTURES AND METHODOLOGIES

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate on graph-structured data. Unlike traditional neural networks, which assume a fixed input size and independent data points, GNNs can naturally incorporate **node features, edge features, and graph topology**. They iteratively aggregate information from a node's neighbors, enabling the network to learn complex dependencies and relational patterns.

Formally, the GNN framework is based on a **message-passing paradigm**, where nodes exchange information with their neighbors and update their representations over multiple layers. Different GNN architectures vary primarily in how they aggregate neighbor information, handle attention, or integrate relational knowledge.

3.1 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs), introduced by Kipf and Welling (2017), are one of the foundational architectures in graph deep learning. They extend the idea of convolution from regular grids (e.g., images) to irregular graph structures.

The **node update rule** in a GCN layer is:

hv(l+1)=σ(∑u∈N(v)1dvduW(l)hu(l))h_v^{(l+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} W^{(l)} h_u^{(l)} \right)hv(l+1)=σu∈N(v)∑dvdu1W(l)hu(l)

Where:

- hv(l)h_v^{(l)}hv(l) is the feature vector of node vvv at layer lll.
- W(l)W^{(l)}W(l) is the learnable weight matrix for layer lll.
- N(v)\mathcal{N}(v)N(v) denotes the set of neighbors of node vvv.
- dvd_vdv and dud_udu are the degrees of nodes vvv and uuu, used for normalization.
- σ\sigmaσ is a non-linear activation function (e.g., ReLU).

Key characteristics of GCNs:

- Aggregates information from neighbors while considering node degree for normalization.
- Supports **semi-supervised learning**, allowing training on partially labeled nodes.
- Can produce **graph-level embeddings** by pooling node features.

Applications:

- Node classification in citation networks (e.g., classifying scientific papers).
- Graph embedding for link prediction or community detection.
- Molecular property prediction in chemical graphs.

Limitations:

- Fixed neighborhood aggregation may treat all neighbors equally, ignoring their relative importance.
- Deep GCNs suffer from **over-smoothing**, where node embeddings converge to similar vectors.

3.2 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) address the limitations of uniform neighbor aggregation by introducing **attention mechanisms**, allowing the network to assign different weights to neighbors based on their relevance.

The **update equation** is:

hv(l+1)=σ(∑u∈N(v)αvuWhu(l))h_v^{(l+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} W h_u^{(l)} \right)hv(l+1)=σu∈N(v)∑αvuWhu(l)

Where αvu\alpha_{vu}αvu is the **attention coefficient**, learned dynamically during training to represent the importance of neighbor uuu to node vvv.

Key features of GATs:

- **Adaptive weighting:** Important neighbors contribute more to the node update.
- **Multi-head attention:** Combines multiple attention mechanisms to stabilize training and enhance expressivity.
- **Interpretability:** Attention scores indicate which neighbors influence predictions.

Applications:

- Social network analysis, highlighting influential users.
- Knowledge graph reasoning, focusing on relevant relationships.
- Traffic and spatial-temporal predictions, identifying key nodes in transportation or sensor networks.

Limitations:

- Higher computational cost due to attention calculations.
- May overfit on small graphs if the attention mechanism is too expressive.

3.3 Message Passing Neural Networks (MPNNs)

Message Passing Neural Networks (MPNNs) provide a **generalized framework** that encompasses GCNs, GATs, and other GNN variants. MPNNs decouple **message computation** and **node update** into two distinct functions:

mv(l+1)=∑u∈N(v)M(hv(l),hu(l),evu)m_v^{(l+1)} = \sum_{u \in \mathcal{N}(v)} M(h_v^{(l)}, h_u^{(l)}, e_{vu})mv(l+1)=u∈N(v)∑M(hv(l),hu(l),evu)

hv(l+1)=U(hv(l),mv(l+1))h_v^{(l+1)} = U(h_v^{(l)}, m_v^{(l+1)})hv(l+1)=U(hv(l),mv(l+1))

Where:

- MMM is the **message function**, which computes messages from neighbor nodes, optionally using edge features evue_{vu}evu.
- UUU is the **update function**, which updates node representations based on aggregated messages.
- mv(l+1)m_v^{(l+1)}mv(l+1) is the aggregated message for node vvv.

Key characteristics:

- Highly flexible: Can handle node, edge, and graph-level features.
- Supports various **message and update functions**, including neural networks, summation, or max-pooling.
- Widely used in chemistry and biology for **molecular property prediction**, modeling interactions between atoms or proteins.

Applications:

- Drug discovery and molecular graph analysis.
- Protein-protein interaction networks.
- Physical simulations of particle systems.

3.4 Relational Graph Neural Networks (R-GNNs)

Relational Graph Neural Networks (R-GNNs) extend GNNs to **multi-relational graphs**, such as knowledge graphs, where edges have different types or semantics. Instead of using a single

weight matrix, R-GNNs maintain **relation-specific transformation matrices** for each edge type:

hv(l+1)=σ(∑r∈R∑u∈Nr(v)Wrhu(l))h_v^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} W_r h_u^{(l)} \right)hv(l+1)=σr∈R∑u∈Nr(v)∑Wrhu(l)

Where:

- R\mathcal{R}R is the set of relation types.
- Nr(v)\mathcal{N}_r(v)Nr(v) are neighbors of node vvv connected by relation rrr.
- WrW_rWr is the learnable transformation matrix for relation rrr.

Key features of R-GNNs:

- Can **differentiate edge types**, capturing richer relational patterns.
- Useful for **link prediction** in knowledge graphs.
- Supports **reasoning over heterogeneous graphs**, where multiple types of nodes and edges coexist.

Applications:

- Knowledge graph completion (e.g., predicting missing relationships).
- Recommender systems with multi-relational data (e.g., user-item interactions with multiple behaviors).
- Social network analysis with multiple interaction types (likes, comments, shares).

4. RECENT ADVANCES IN GNNS & RELATIONAL ML

Graph Neural Networks (GNNs) and Relational Machine Learning (Relational ML) have evolved significantly, driven by the need to handle **large-scale graphs**, **heterogeneous relational structures**, and **self-supervised learning paradigms**. This section highlights the recent methodological and application-oriented advances that have enhanced the scalability, expressivity, and applicability of graph-based learning.

4.1 Scalability Techniques

Large graphs, such as social networks or protein interaction networks, can contain millions of nodes and billions of edges. Naïve GNN implementations require full-batch training on the adjacency matrix, which is computationally prohibitive. Recent advances focus on **scalable GNN methods** that reduce memory requirements while maintaining predictive performance.

Key techniques include:

1. **Graph Sampling:**
    - Methods like **GraphSAGE** and **Cluster-GCN** perform neighborhood or cluster-based sampling instead of using the full adjacency matrix.
    - GraphSAGE samples a fixed-size neighborhood for each node during training, enabling **inductive learning** on unseen nodes.
    - Cluster-GCN divides the graph into clusters and performs mini-batch training on subgraphs, preserving local connectivity.
    - Sampling reduces computational load and allows GNNs to scale to large networks without sacrificing representation quality.
1. **Mini-batch Training:**
    - Full-graph training can exhaust GPU memory. Mini-batch methods process **subsets of nodes and edges**, updating model weights incrementally.
    - Coupled with sampling, mini-batch training improves **convergence speed** and enables **online learning** in dynamic graphs.
2. **Sparse Matrix Operations:**
    - Many real-world graphs are sparse, meaning most node pairs are unconnected.
    - Leveraging **sparse adjacency matrices** reduces computational complexity from O(N2)O(N^2)O(N2) to O(E)O(E)O(E), where EEE is the number of edges.
    - Libraries like **PyTorch Geometric** and **DGL** implement sparse tensor operations optimized for GPU acceleration.

Impact: Scalability techniques have made GNNs applicable to **web-scale networks**, large knowledge graphs, and industrial datasets, enabling real-time inference in recommender systems and fraud detection.

4.2 Self-Supervised and Contrastive GNNs

Labeling graph data is expensive, especially in domains like biology or knowledge graphs. **Self-supervised learning (SSL)** provides a solution by creating **pseudo-labels from graph structure**, allowing GNNs to learn meaningful embeddings without full supervision.

Self-supervised approaches:

1. **Node-level SSL:**
    - Techniques like **GraphMAE** mask node features and train the model to reconstruct

them, similar to masked language modeling in NLP.

- Useful for **node classification** and downstream tasks when labeled data is scarce.

2. **Edge-level SSL:**
   - Predicts the existence or type of edges between nodes.
   - Improves **link prediction** and knowledge graph completion.
3. **Graph-level SSL (Contrastive Learning):**
   - Methods like **GraphCL** create two augmented views of the same graph and maximize the agreement between their embeddings.
   - Enables **unsupervised graph representation learning** for tasks like graph classification, molecular property prediction, and recommendation.

Benefits:

- Reduces reliance on labeled data.
- Produces **robust embeddings** that generalize well across tasks.
- Integrates naturally with downstream relational ML tasks.

4.3 Integration with Relational Machine Learning

GNNs and Relational ML can be **combined** to leverage both **statistical relational learning** and **graph-based deep learning**, enabling advanced reasoning over structured knowledge.

Techniques and applications:

1. **Knowledge Graph Embeddings:**
   - Traditional methods (e.g., TransE, DistMult) represent entities and relations in a latent space.
   - GNN-based embeddings incorporate **graph topology**, neighbor context, and multi-hop relational information, producing richer representations.

2. **Link Prediction:**
   - Multi-relational graphs often contain **missing edges** (e.g., unknown interactions in protein networks).
   - GNNs can aggregate neighborhood information across multiple relation types, predicting **probable links** more accurately than shallow factorization methods.

3. **Rule-Guided Learning:**
    - Logical rules can be encoded to **guide GNN training**, improving interpretability and generalization.
    - Example: In a social network, a rule like "if A and B are friends, and B and C are friends, A is likely to know C" can be embedded as a regularization term in GNN loss functions.

Impact: Integration enhances **predictive performance**, **reasoning capabilities**, and **interpretability**, bridging symbolic and neural approaches.

4.4 Multi-Relational & Heterogeneous GNNs

Many real-world graphs are **heterogeneous**, containing multiple types of nodes and edges. Multi-relational GNNs and heterogeneous GNNs have been proposed to handle this complexity.

Key architectures:

1. **Relational GCN (R-GCN):**
    - Introduces **relation-specific weight matrices**, allowing the model to treat each edge type differently.
    - Supports tasks like **link prediction** and **entity classification** in knowledge graphs (e.g., Freebase, Wikidata).
2. **Heterogeneous Graph Attention Network (HAN):**
    - Combines **node- and edge-type attention mechanisms** to selectively aggregate information from heterogeneous neighbors.
    - Useful for recommendation systems where nodes may represent users, items, or categories, and edges represent different interaction types (click, view, purchase).
3. **Heterogeneous Graph Transformers:**
    - Utilize **transformer-based attention** for long-range dependencies and multi-relational reasoning.
    - Effective in citation networks, academic recommendation systems, and biomedical knowledge graphs.

Challenges addressed:

- Heterogeneous graphs often suffer from **imbalanced node types** and **sparse edge relations**.

- Multi-relational GNNs can differentiate between relations, improving **prediction accuracy** and **embedding quality**.

5. APPLICATIONS

5.1 Social Network Analysis

GNNs model user interactions for:

- Community detection
- Influence prediction
- Fake account detection

5.2 Recommender Systems

GNNs exploit user-item graphs for collaborative filtering, outperforming traditional matrix factorization approaches.

5.3 Bioinformatics & Chemistry

- Protein–protein interaction prediction
- Drug discovery via molecular graphs
- Gene regulatory networks

5.4 Knowledge Graph Completion

GNNs aid in link prediction, entity classification, and reasoning over complex multi-relational knowledge graphs.

CHALLENGES AND LIMITATIONS

- **Scalability:** Large-scale graphs require efficient sampling and memory management.
- **Over-smoothing:** Deep GNN layers may converge to indistinguishable node embeddings.
- **Interpretability:** Understanding learned embeddings and decisions is non-trivial.
- **Heterogeneity:** Diverse node and edge types complicate learning.
- **Dynamic graphs:** Temporal changes require online or incremental learning strategies.

FUTURE DIRECTIONS

- **Integration with symbolic AI:** Combining logic and GNNs for explainable reasoning.
- **Temporal GNNs:** Modeling evolving graphs in social networks and finance.
- **Graph generative models:** Designing molecules, circuits, or synthetic networks.
- **Multi-modal graph learning:** Combining textual, visual, and structured graph data.

CONCLUSION

Graph Neural Networks and Relational Machine Learning provide a unified framework to learn from structured and relational data, overcoming the limitations of conventional models. GNNs excel in capturing node features, edge relationships, and global structures, while relational ML incorporates dependencies and probabilistic reasoning. Despite challenges in scalability, interpretability, and dynamic graph modeling, recent advancements have expanded applications across social networks, recommender systems, bioinformatics, and knowledge graphs. Future research should emphasize integration with symbolic reasoning, temporal graph modeling, and multi-modal learning to unlock the full potential of relational and graph-based AI.

REFERENCES

1. Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *ICLR 2017*.
2. Velickovic, P., et al. (2018). Graph Attention Networks. *ICLR 2018*.
3. Gilmer, J., et al. (2017). Neural Message Passing for Quantum Chemistry. *ICML 2017*.
4. Schlichtkrull, M., et al. (2018). Modeling Relational Data with Graph Convolutional Networks. *ESWC 2018*.
5. Hamilton, W., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *NeurIPS 2017*.
6. Wang, X., et al. (2019). Heterogeneous Graph Attention Network. *WWW 2019*.
7. Nickel, M., et al. (2016). A Review of Relational Machine Learning for Knowledge Graphs. *Proc. of IEEE*.
8. Battaglia, P., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*.
9. Wu, Z., et al. (2020). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*.
10. Rong, Y., et al. (2020). Self-Supervised Graph Representation Learning: A Survey. *arXiv:2009.03509*.