Graph Similarity Computation Using Convolution Neural network

Ashwani Kumar Yadav, Prakash Dwivedi

Department of Information Technology G. B. Pant University of Agriculture and Technology Pantnagar, India

Email: ashwaniittalks@gmail.com

ABSTRACT

Graph similarity computation is a foundational problem with broad applications including molecule comparison, social network analysis, bioinformatics, and code analysis. Traditional methods such as GED and MCS are NP-hard and impractical for large graphs. The emergence of graph neural networks (GNNs), particularly convolution-based architectures has revolutionized graph similarity learning by leveraging deep learning techniques for efficient and scalable prediction. This paper provides a comprehensive overview of convolutional neural network (CNN)–based architectures for graph similarity computation, emphasizing their scalability, interpretability, and adaptability. The discussion extends to graph autoencoders (GAE), generative adversarial networks (GAN), and hybrid frameworks that learn structural embeddings capable of generalizing across domains. We analyze key models including SimGNN, GraphSim, GSimCNN, and SEGMN, highlighting their mechanisms for multi-scale feature extraction, convolutional matching, and structure-enhanced reasoning. This review comprehensively examines convolutional architectures for graph similarity computation, including mathematical foundations, CNN-based set matching, node ordering schemes, applications, challenges and future research directions.

Keywords: *Graph Similarity Learning; Graph Neural Networks (GNNs); Convolutional Neural Networks (CNNs); Graph Matching; Graph Edit Distance (GED); Graph Representation Learning; Structure-Enhanced Networks; Deep Learning*.

1. Introduction

Graph similarity learning is vital across various domains such as chemistry, bioinformatics, recommender systems, and social networks [1], [12]. Traditional similarity measures—Graph Edit Distance (GED) and Maximum Common Subgraph (MCS) are computationally intractable for large graphs [10], [11]. The rise of deep learning, especially convolutional graph representation learning, has enabled practical and powerful graph similarity computation methods [6]. Measuring similarity across graphs enables core operations such as similarity search, database analytics, clustering, and anomaly detection, while supporting downstream tasks like molecule comparison, protein interaction analysis, community analysis, malware detection, and recommendation via structural alignment. Exact GED/MCS computation is NP-hard and intractable on large graph; machine learning-based methods aim to approximate these metrics from labeled graph pairs, enabling fast prediction on unseen pairs with GPU acceleration [3], [4].

Graph similarity learning aims to measure how structurally and semantically alike two graphs are. Traditional methods such as graph kernels or edit distances are computationally expensive, especially for large graphs [10],[20]. Deep learning–based methods, particularly Graph Convolutional Networks (GCN), Graph Autoencoders (GAE), and Generative Adversarial

Networks (GAN), provide efficient and scalable alternatives by learning low-dimensional embeddings that capture graph structure and features [6], [7], [8], [13].

A Graph Convolutional Network (GCN) extends convolution operations to graph-structured data [6], [15]. Instead of using pixel neighborhoods as in images, a GCN aggregates information from each node's local neighborhood based on the graph's adjacency matrix. Mathematically, the propagation rule for a single GCN layer is expressed as:

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

where A is the adjacency matrix, I the identity matrix (for self-loops), $W^{(l)}$ the layer's weight matrix, and σ a nonlinear activation function [6], [15]. Each layer updates node embeddings by combining features from neighbouring nodes. After several layers, a global pooling operation aggregates node embeddings into a single vector representing the entire graph. For graph similarity tasks, the cosine similarity or Euclidean distance between these graph-level embeddings can quantify how similar two graphs are [3], [5].

A Graph Autoencoder (GAE) is an unsupervised neural model that learns graph representations by reconstructing the graph structure from latent embeddings [7]. It consists of two components: an encoder and a decoder. The encoder, often built using GCN layers, generates embeddings $Z = \mathrm{GCN}(A, X)$ that capture the node's structural relationships and attributes. The decoder reconstructs the adjacency matrix using $\hat{A} = \sigma(ZZ^T)$, where Z are the learned embeddings. The reconstruction loss, typically the Frobenius norm $\| A - \hat{A} \|^2$ forces the encoder to learn embeddings that preserve graph connectivity [7], [8]. These embeddings can then be compared between graphs to assess similarity. GAEs are especially useful when labeled similarity data is unavailable, as they learn in an unsupervised manner.

A Generative Adversarial Network (GAN) further enhances graph similarity learning by introducing an adversarial training process [13]. It comprises a generator (G) and a discriminator (D) engaged in a minimax game. The generator learns to produce realistic graph embeddings or synthetic graphs, while the discriminator tries to distinguish between real and generated samples. Through this adversarial process, the generator learns to produce embeddings that closely match the distribution of real graph data [8], [13]. In graph similarity tasks, GANs regularize the embedding space, ensuring that graphs with similar structures have nearby embeddings and dissimilar graphs are well separated.

Convolutional set matching methods— Graph Similarity Computation via CNNs (GSimCNN) [6] and GraphSim [4] eschew single graph-level embeddings in favour of a CNNs operating on node-node similarity matrices, capturing fine-grained structural correspondences and multi-scale interactions that fixed-size graph embeddings miss. Structure-enhanced graph matching Network (SEGMN) broadens this paradigm with dual node-edge embeddings and assignment-graph convolution to inject cross-graph structural context into matching, further stabilizing similarity estimates [4],[14].

This paper is organized into six sections that collectively examine the development and impact of convolutional neural network–based graph similarity computation. The Introduction outlines the motivation and contrasts traditional edit-distance and kernel-based approaches [33] with modern deep learning models such as GCNs, GAEs, and GANs. The Background section

explains the fundamentals of graph representation and embedding methods. The Convolutional Architectures section reviews major CNN-based models—SimGNN, GraphSim, GSimCNN, and SEGMN with emphasis on their design and comparative performance. The Applications section highlights the use of graph similarity learning in chemistry, biology, social networks, software engineering, and recommender systems. The Challenges and Future Directions discuss scalability, interpretability, and generalization issues, outlining paths toward more efficient and explainable models. Finally, the Conclusion summarizes key insights and underscores the growing importance of structure-aware convolutional approaches in graph similarity learning.

2. **Background: Graph Similarity Foundations**

A graph is a mathematical structure represented as $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ denotes a set of nodes (or vertices), and $E \subseteq V \times V$ represents the set of edges connecting pairs of nodes[16]. Graphs are versatile representations that capture relational dependencies found in various domains, including molecular chemistry, social networks, and knowledge graphs[1],[12],[17].

The adjacency matrix $A \in \mathbb{R}^{n \times n}$ encodes the connectivity of a graph, where $A_{ij} = 1$ if an edge exists between nodes v_i and v_j, and $A_{ij} = 0$ otherwise. For weighted graphs, A_{ij} stores the corresponding edge weight. This matrix serves as the primary input for most graph neural networks, determining how information propagates between connected nodes[6],[15]. To ensure proper normalization during learning, the degree matrix D is introduced, which is a diagonal matrix with entries $D_{ii} = \sum_j A_{ij}$, representing the number of edges incident to each node[18].

The graph Laplacian, defined as $L = D - A$, provides a compact representation of graph structure and is essential in spectral graph theory [18]. The normalized Laplacian, $L_{norm} = I - D^{-1/2} A D^{-1/2}$, is frequently used to stabilize learning in graph convolutional models by removing scale dependency among nodes [6],[18]. These matrix representations enable graph-based neural models to perform convolution-like operations that generalize traditional signal processing to irregular graph domains [6],[15].

Nodes in a graph may also have associated features or attributes represented by a feature matrix $X \in \mathbb{R}^{n \times d}$, where each row X_i corresponds to the feature vector of node v_i and d denotes the feature dimension. These features can encode physical, semantic, or statistical properties, such as atomic characteristics in molecules or demographic traits in social networks[1],[12].

A key concept in graph representation learning is the node embedding, where each node v_i is mapped to a dense vector $z_i \in \mathbb{R}^k$ in a low-dimensional space[12],[15]. The goal is to preserve the structural and semantic relationships of the graph, such that nodes with similar roles or neighborhoods have embeddings close to each other. To represent an entire graph, a graph-level embedding h_G is obtained by aggregating all node embeddings using a readout function, such as sum, mean, max pooling, or attention-based aggregation[6],[15]:

$$h_G = \text{READOUT}(\{z_1, z_2, \dots, z_n\})$$

This embedding captures global graph properties and serves as the input for similarity computation [3],[5]. The graph similarity measure quantifies the resemblance between two graphs, $G_1$and G_2, using their embeddings:

$$\text{Sim}(G_1, G_2) = f(h_{G_1}, h_{G_2})$$

where $f$can be cosine similarity, dot product, or Euclidean distance [3],[5]. Higher similarity values indicate that the graphs share structural or functional properties.

Two graphs are said to be isomorphic if there exists a one-to-one mapping between their node sets that preserves edge connections. Formally, $G_1 = (V_1, E_1)$and $G_2 = (V_2, E_2)$are isomorphic if there exists a bijection $f: V_1 \to V_2$such that $(v_i, v_j) \in E_1 \Leftrightarrow (f(v_i), f(v_j)) \in E_2$ [16],[19]. Although exact isomorphism is computationally challenging, modern learning-based models approximate it through embedding similarity [3],[4],[5].
Finally, message passing forms the backbone of graph neural networks, where each node iteratively aggregates information from its neighbours to update its representation [6],[15]:

$$h_i^{(k)} = \text{UPDATE}(h_i^{(k-1)}, \text{AGGREGATE}(\{h_j^{(k-1)} : j \in \mathcal{N}(i)\}))$$

This process enables multi-hop information propagation, allowing the model to capture higher-order relationships essential for graph similarity learning. Together, these preliminaries establish the theoretical foundation for advanced graph-based deep learning models that assess structural resemblance efficiently and accurately [1],[12].
Graph similarity learning aims to determine how similar two graphs are in structure and semantics [3],[5]. Early approaches such as GED [30],[31] and Graph Kernels [33] relied on combinatorial or feature-based comparisons [10],[11],[20]. GED measures the minimum number of edit operations (additions, deletions, substitutions) needed to transform one graph into another, providing an intuitive notion of similarity but suffering from high computational cost [10],[11]. Graph kernels, such as the Weisfeiler–Lehman (WL) kernel, Shortest Path kernel, and Graphlet kernel, compare substructures like paths or motifs between graphs [20],[21]. Although effective, these methods often struggle with scalability.
To overcome these limitations, GNN based methods have become dominant [6],[12],[15]. GCNs extend convolution to irregular graph domains by aggregating neighbourhood information, producing node and graph embeddings that capture both local and global structure [6]. Siamese GNNs employ two identical encoders with shared weights to produce embeddings for two graphs, and their similarity is measured using a distance or contrastive loss [3],[5]. Graph Matching Networks (GMNs) enhance this by allowing cross-graph attention, learning fine-grained node correspondences [14].
GAEs and Variational Graph Autoencoders (VGAEs) represent unsupervised approaches that learn embeddings by reconstructing the adjacency matrix [7],[8]. GAEs preserve the topological structure, while VGAEs introduce probabilistic latent spaces for smoother embeddings, enabling similarity computation in the latent domain.

GANs further refine embedding learning by introducing a generator–discriminator framework that enforces realistic and discriminative representations [13]. Contrastive learning techniques, such as GraphCL and InfoGraph, use positive (similar) and negative (dissimilar) graph pairs to directly optimize for similarity-preserving embeddings [22],[23]. Modern systems often integrate these methods—for example, combining GCNs, GAEs, and GANs—to achieve robust and scalable graph similarity estimation [3],[4],[13]. These advanced deep learning approaches have become essential for applications in chemistry, bioinformatics, and social network analysis [1],[12].

3. **Convolutional Architectures**

Our focus in this section is on convolution-based *pairwise* frameworks rather than single-graph encoders. Convolution-based pairwise frameworks represent a major shift from traditional single-graph encoders by directly modelling interactions between two input graphs. These architectures learn cross-graph representations that jointly encode structure and node correspondence, offering scalable and differentiable alternatives [23],[25],[26] to combinatorial matching methods such as graph edit distance (GED) [10], [11].

(a)**SimGNN:**

SimGNN is a two-stage convolutional model designed for end-to-end graph similarity computation [3]. It first generates node embeddings using a GCN and then employs a Neural Tensor Network (NTN) and histogram matching module to compute similarity between embedded graphs.

$$\text{Sim}(G_1, G_2) = \text{MLP}([NTN(z_1, z_2); hist(S)])$$

This approach captures both structural and distributional similarity: the NTN models pairwise node interactions, while the histogram encodes global matching distributions. Its strength lies in combining fine-grained correspondence with global comparison; however, it requires post-embedding alignment and scales poorly for large graphs [3], [5].

(b) **GraphSim**

GraphSim extends SimGNN by integrating multi-scale convolutional set matching, learning hierarchical correspondences between node sets at different resolutions [4]. It employs attention-based convolutional layers to generate multi-resolution representations, allowing the model to capture both local and global structures effectively. Convolutional matching, on the other hand, provides an alignment-free and fully differentiable approach [26] through the use of learned filters, allowing the network to automatically extract and compare relational features between graphs without the need for explicit node correspondence. GraphSim thus enhances robustness and performance on standard GED benchmarks (e.g., AIDS, LINUX, and IMDB), achieving superior rank correlation and mean squared error over prior models [4].

(c) **Graph Matching Network**

Graph Matching Networks (GMNs) introduce cross-graph message passing to learn node correspondences directly [14]. During propagation, each node in graph G_1 updates its representation by attending to all nodes in G_2:

$$h_i^{G_1} \leftarrow f(h_i^{G_1}, \{h_j^{G_2}\})$$

This attention-guided update mechanism allows information exchange across graphs, enabling the model to learn fine-grained structural alignments. GMNs are particularly effective for subgraph isomorphism and graph edit distance approximation, outperforming simpler Siamese frameworks in capturing relational consistency [14].

(d) **Graph Siamese Convolution Architecture:**

Siamese convolutional architectures formulate graph similarity as a pairwise metric learning problem using shared-parameter encoders [6], [12]. A typical formulation is:

$$s(G_1, G_2) = f_\theta(\phi(G_1), \phi(G_2))$$

where ϕ is a shared GCN/GAT encoder and f_θ represents a similarity function such as a multilayer perceptron (MLP) or cosine similarity. Its variants include Graph Matching Siamese GCN, DGMN, and SimGNN-Lite. These architectures offer parameter efficiency, symmetric training, and fast inference, making them well-suited for large-scale graph similarity search.

(e) **Deep learning-based frameworks**:

Deep learning frameworks recast graph similarity as a pattern recognition task over node embeddings or pairwise similarity matrices. Convolutional similarity learning extends message passing's locality principle by using CNNs to extract translationally invariant motifs from similarity "images." This approach parallels optimal transport and bipartite assignment methods but replaces explicit optimization with learned convolutional filters [4], [6]. Graph generative models, such as GraphRNN [24], further inspire Breadth-First Search (BFS) canonicalization, ensuring consistent node ordering for CNN-based processing and establishing a methodological link between graph generation and comparison pipelines. Modern GNN surveys classify these models within ConvGNNs for graph-level tasks but highlight node-level interactions as critical for fine-grained similarity prediction [1], [12].

(f) **Graph Similarity Computation via CNNs**

GSimCNN [6] generates node embeddings through multi-scale GCN layers, then constructs pairwise node–node similarity matrices. These matrices are resized via bilinear interpolation and fed into 2D CNNs followed by an MLP regression head to predict similarity scores. This approach converts the NP-hard graph similarity estimation problem—often defined via Graph Edit Distance (GED)—into a learnable regression problem [2], [3]. GSimCNN achieves high parallelizability and efficiency due to its neural computation framework, outperforming traditional GED approximators while maintaining interpretability [4].

(g) GraphSim (Multi-scale Matching CNNs)

GraphSim [4] builds upon GSimCNN by introducing multi-resolution similarity matrices and CNN-based feature extraction to identify relational motifs. It achieves state-of-the-art performance on GED and maximum common subgraph (MCS) tasks across multiple benchmark datasets [34]. Unlike fixed embedding models, GraphSim performs direct node-level matching, enhancing robustness, scalability, and accuracy by explicitly encoding hierarchical similarity patterns.

(h) Structure-Enhanced Graph Matching Network: SEGMN

The Structure-Enhanced Graph Matching Network (SEGMN) [5] introduces two innovations that significantly advance convolutional similarity learning:

a) Dual Embedding Construction:
SEGMN constructs a line graph to encode edge information, applying edge-enhanced GCNs to generate edge embeddings. These are concatenated with degree-weighted aggregates of adjacent edge embeddings to form dual node–edge representations, improving structure fidelity, especially for unlabeled graphs where topology dominates GED performance [5].
b) Assignment-Graph Convolution:
SEGMN builds an assignment graph, where nodes represent cross-graph node pairs. Convolution operations propagate similarity among structurally related pairs, correcting mismatches through cross-graph structural constraints. This plug-and-play module enhances multiple baselines (e.g., GraphSim, SimGNN) by notable margins [5].

By integrating these components, SEGMN introduces global pairwise structure perception, addressing a key limitation of intra-graph message passing, which traditionally lacks explicit modeling of cross-graph relations [5].

4. Applications of Graph Similarity

Graph similarity learning has emerged as a crucial paradigm across numerous domains where data is naturally represented as graphs. By leveraging graph neural networks (GNNs), autoencoders, and convolutional architectures, these methods can effectively learn structural embeddings that preserve both node-level and global graph information. The following are key application areas where graph similarity learning plays an important role:

1. **Chemoinformatics:**
In chemical and pharmaceutical research, molecules are often modelled as graphs, with atoms as nodes and chemical bonds as edges. Graph similarity learning enables similarity search, lead optimization, and drug discovery by comparing molecular structures efficiently. Traditional measures such as GED or MCS provide theoretical similarity but are computationally expensive[29], [30]. Deep learning–based methods use GNNs as surrogates for GED or MCS, significantly accelerating molecular comparison. Convolutional approaches also preserve substructure sensitivity, which is critical for identifying key functional groups that determine Structure–Activity Relationships (SAR) in drug discovery and toxicology prediction.

2. **Bioinformatics:**
   In biological networks such as protein–protein interaction (PPI) and gene regulatory networks, graph similarity learning is used to analyze functional and structural similarity among biological entities [7]. Proteins with similar graph topologies or recurring motifs often exhibit similar biological functions. By learning topological embeddings, GNN-based similarity models facilitate protein classification, disease–gene association prediction, and functional annotation transfer. Such representations help biologists understand evolutionary relationships and identify novel interactions[39].

3. **Social_Networks:**
   In social network analysis, individuals or groups can be represented as nodes connected by social interactions [35]. Graph similarity learning aids in identifying community-level similarities, user behavior matching, and cross-network alignment. For instance, it can be used to match user profiles across different platforms (e.g., Twitter and Facebook) or detect similar interest-based communities. By capturing relational patterns and connectivity, these methods enhance social recommendations, influence analysis, and fraud detection.

4. **Software Analysis:**
   Graphs are also widely used to represent software structures, such as control-flow graphs (CFGs) and program dependency graphs (PDGs)[32]. Graph similarity learning supports binary function similarity detection, malware analysis, and code plagiarism detection by comparing structural patterns within code graphs [32]. Neural graph matching models enable scalable comparison of program structures, identifying semantically equivalent code segments even with syntactic variations.

5. **Recommendation Systems:**
   In recommendation scenarios, items and users can be represented as nodes in a knowledge graph, where edges encode semantic relationships[37]. Graph similarity learning measures item–item similarity through structural proximity rather than relying solely on explicit attributes. This helps address the cold-start problem when user or item attributes are sparse[37],[23]. By leveraging learned graph embeddings, these systems enhance personalized recommendations, link prediction, and content discovery.

Overall, graph similarity learning provides a unifying framework that enhances performance across chemistry, biology, social sciences, cybersecurity, and recommendation systems by modeling relational structures and enabling efficient similarity computation.

5. Challenges and Future Directions

Despite significant progress, graph similarity learning still faces several open challenges that limit its scalability, interpretability, and adaptability across diverse domains. Addressing these issues is crucial for developing more generalizable and explainable models capable of handling dynamic and heterogeneous graph data. One of the foremost challenges is scalability. Real-world graphs such as molecular, social, and knowledge networks often consist of millions of nodes and edges, making pairwise similarity computation computationally prohibitive. The formation of dense similarity matrices introduces massive memory and processing overhead. Approaches such as efficient sampling, graph sparsification, and hierarchical coarsening have been proposed to reduce this complexity by compressing large graphs into smaller,

representative structures. Frameworks like GraphSAGE and Graph Hierarchical Networks partially mitigate this by processing localized subgraphs or clusters; however, achieving true scalability for web-scale or industrial datasets remains an open research goal.

Another major issue is interpretability. While convolutional [28] and attention-based graph models[27] achieve strong similarity accuracy, they largely function as "black boxes," offering limited transparency into which substructures or node attributes drive similarity predictions [4],[5]. Convolutional filters implicitly capture motifs contributing to similarity, but tracing these learned patterns back to meaningful graph components remains difficult. Recent advances such as assignment-graph convolutions and explainable GNN frameworks attempt to map embeddings back to interpretable subgraphs, yet standardized explanation protocols and benchmarks for structural attribution are still lacking in graph similarity learning[40].

The heterogeneity and attribute diversity of real-world graphs introduce additional challenges. Many graphs involve multiple node and edge types, complex relationships, and high-dimensional features. Traditional similarity models that focus solely on unlabelled graph structures fail to capture such semantic richness. Future approaches must incorporate edge-aware and relation-specific embeddings along with cross-graph attention mechanisms to effectively model multi-relational graphs. Integrating heterogeneous features—including textual, numerical, and categorical modalities into unified graph embeddings is an ongoing area of investigation.

Another dimension of complexity arises from graph dynamics. Most current methods assume static graph structures, whereas many real systems evolve continuously over time. Learning similarity over dynamic graphs requires modelling sequences of time-stamped graphs or evolving adjacency matrices. Temporal GNNs and spatiotemporal similarity networks offer promising solutions by capturing how structural similarity changes as the graph evolves—an ability essential for applications such as fraud detection, molecular interaction analysis, and social trend forecasting.

Finally, domain shift represents a critical barrier to generalization. Models trained on chemical or biological datasets often fail to transfer effectively to social or program graphs due to variations in structural distributions and attribute semantics. Techniques such as self-supervised pretraining, contrastive learning, and meta-learning can help align representations across domains by leveraging structural invariances. The development of universal graph encoders capable of domain-agnostic similarity estimation remains a promising direction for the field.

Overall, overcoming these challenges through scalable architectures, interpretable learning mechanisms, and robust cross-domain generalization strategies will define the next generation of graph similarity learning models moving the field toward more efficient, explainable, and transferable graph understanding.

6. Conclusion

Convolutional neural network approaches have reshaped graph similarity learning by converting node embedding interactions into images that CNNs can efficiently parse, achieving high-fidelity approximations of GED/MCS with strong runtime advantages over exact solvers. Structure-enhanced frameworks incorporate edge-aware dual embeddings and assignment-graph reasoning to better capture cross-graph structural constraints, improving accuracy and robustness on labeled and unlabeled benchmarks. Going forward, hybrid attention-convolution models, self-supervised pretraining, differentiable assignment reasoning, and interpretable structure-aware matching will drive further advances and broaden applicability across scientific and industrial graph similarity tasks.

References

[1]. P. Riba, A. D. Bagdanov, D. Karatzas, and M. Rusiñol, "Learning Graph Distances with Message Passing Neural Networks," Proc. IEEE Int. Conf. Pattern Recognit. (ICPR), 2018.

[2]. Y. Bai, H. Ding, Y. Sun, and W. Wang, "Convolutional Set Matching for Graph Similarity," Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2018.

[3]. Y. Bai, H. Ding, K. Gu, Y. Sun, and W. Wang, "Learning-Based Efficient Graph Similarity Computation via Multi-Scale Convolutional Set Matching," Proc. AAAI Conf. Artif. Intell., 2020.

[4]. Y. Bai et al., "SimGNN: A Neural Network Approach to Fast Graph Similarity Computation," Proc. ACM Int. Conf. Web Search Data Mining (WSDM), 2019.

[5]. W. Wang, J. Lu, K. Chen, Z. Liu, and S. Sang, "SEGMN: A Structure-Enhanced Graph Matching Network for Graph Similarity Learning," arXiv preprint arXiv:2411.03624, 2024.

[6]. X. Qin et al., "EGSC-Transformer for Graph Similarity," 2021.

[7]. C. Ling et al., "MGMN: Matching Guided Graph Matching Network," 2021.

[8]. Y. Tan et al., "NA-GSL: Neighborhood-Aware Graph Similarity Learning," 2023.

[9]. A. Roy et al., "ISONET: Edge Alignment Network for Graph Similarity," 2022.

[10]. J. Cho, S. Lee, and K. Lee, "Reweighted Random Walk Matching," 2010.

[11]. L. Yu et al., "Graph Neural Network for Supervised Seeded Graph Matching," Proc. Int. Conf. Machine Learning (ICML), 2023.

[12]. Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph Matching Networks for Learning the Similarity of Graph Structured Objects," Proc. Int. Conf. Machine Learning (ICML), 2019.

[13]. Xiu et al., "Hierarchical Graph Matching Network (HGMN)," 2020.

[14]. Graph Matching Consensus, Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[15]. Y. Sun, X. Yan, T. Wu, J. Han, and P. Yu, "PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks," Proc. VLDB Endowment, 2011.

[16]. Y. Sun et al., "Graph Partitioning and Graph Neural Network Based Similarity Computation," Neurocomputing, 2021.

[17]. Graph-Based Similarity of Deep Neural Networks, ScienceDirect, 2024.

[18]. Joint Graph Learning and Matching (GLAM), ScienceDirect, 2022.

[19]. Neural Graph Matching for Pre-Training Graph Neural Networks, arXiv preprint arXiv:2203.01597, 2022

[20]. Graph Similarity Computation via Interpretable Neural Node Alignment, arXiv preprint arXiv:2412.12185, 2024.

[21]. T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[22]. T. Kipf and M. Welling, "Variational Graph Auto-Encoders," arXiv preprint arXiv:1611.07308, 2016.

[23]. W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017.

[24]. K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?," Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[25]. J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models," Proc. Int. Conf. Machine Learning (ICML), 2018.

[26]. R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical Graph Representation Learning with Differentiable Pooling," Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2018.

[27]. P. Veličković et al., "Graph Attention Networks," Proc. Int. Conf. Learn. Represent. (ICLR), 2018.

[28]. M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2016.

[29]. H. Bunke, "What is the Distance Between Graphs?," Bull. Eur. Assoc. Theor. Comput. Sci. (EATCS), 1983.

[30]. K. Riesen and H. Bunke, "Approximate Graph Edit Distance Computation by Means of Bipartite Graph Matching," Image and Vision Computing, vol. 27, no. 7, pp. 950–959, 2009.

[31]. A. Dabah et al., "Efficient Approximate Approach for Graph Edit Distance," Pattern Recognit. Lett., vol. 145, pp. 125–132, 2021.

[32]. J. Ferrante, K. J. Ottenstein, and J. D. Warren, "The Program Dependence Graph and Its Use in Optimization," ACM Trans. Program. Lang. Syst. (TOPLAS), vol. 9, no. 3, pp. 319–349, 1987.

[33]. P. Yanardag and S. Vishwanathan, "Deep Graph Kernels," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD), 2015.

[34]. C. Morris et al., "TUDataset: A Collection of Benchmark Datasets for Learning with Graphs," arXiv preprint arXiv:2007.08663, 2020.

[35]. Shri Prakash Dwivedi, Ravi Shankar Singh, "Structural Pattern Recognition Using Graph Matching" CRC Press, (2025). [Book DOI: 10.1201/9781003532248].

[36]. D. Bo, C. Shi, L. Wang, and R. Liao, "SpecFormer: Spectral GNNs Meet Transformers," 2023.

[37]. M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and Deep Graph Convolutional Networks," 2020.

[38]. E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive Universal Generalized PageRank Graph Neural Network," 2020.

[39]. C. Hamilton et al., "Protein Graph Representation Learning via Graph Neural Networks," Briefings in Bioinformatics, 2021.

[40]. H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in Graph Neural Networks: A Taxonomic Survey," IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI), 2023.