Database, 2023, 1–20
DOI: https://doi.org/10.1093/database/baad045
Review

The importance of graph databases and graph learning for clinical applications

Daniel Walke[1,2,*,‡], Daniel Micheel[2,‡], Kay Schallert[3], Thilo Muth[4], David Broneske[5], Gunter Saake[2] and Robert Heyer[3,6]

[1]Bioprocess Engineering, Otto von Guericke University, Universitätsplatz 2, Magdeburg 39106, Germany
[2]Database and Software Engineering Group, Otto von Guericke University, Universitätsplatz 2, Magdeburg 39106, Germany
[3]Multidimensional Omics Analyses Group, Leibniz-Institut für Analytische Wissenschaften—ISAS—e.V., Bunsen-Kirchhoff-Straße 11, Dortmund 44139, Germany
[4]Section eScience (S.3), Federal Institute for Materials Research and Testing (BAM), Unter den Eichen 87, Berlin 12205, Germany
[5]Infrastructure and Methods, German Center for Higher Education Research and Science Studies (DZHW), Lange Laube 12, Hannover 30159, Germany
[6]Faculty of Technology, Bielefeld University, Universitätsstraße 25, Bielefeld 33615, Germany

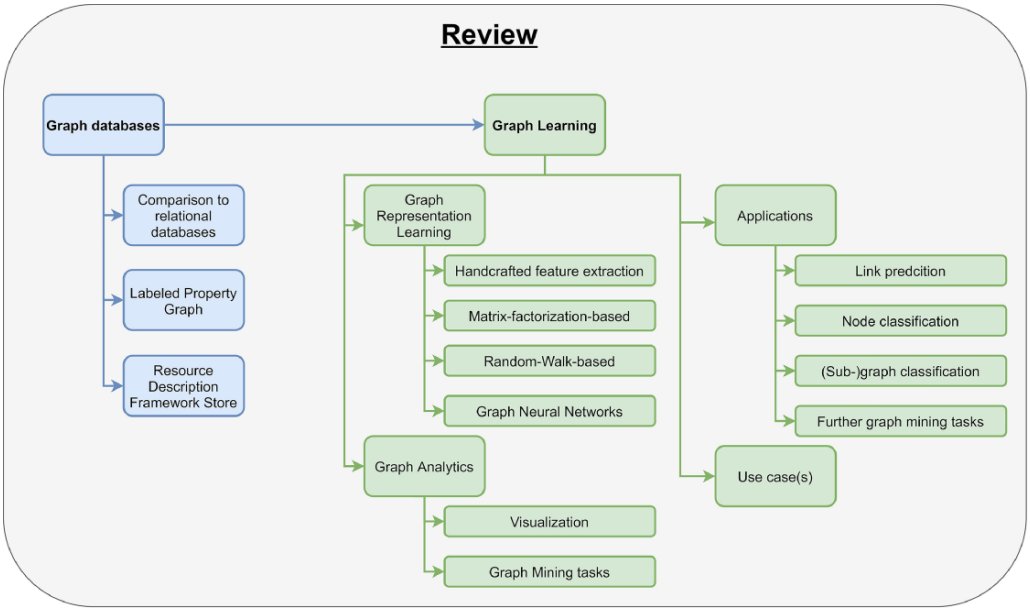
[*]Corresponding author: Tel: +49 391 6752845; Email: daniel.walke@ovgu.de
[‡]First authors.

Citation details: Walke, D., Micheel, D., Schallert, K. *et al.* The importance of graph databases and graph learning for clinical applications. *Database* (2023) Vol. 2023: article ID baad045; DOI: https://doi.org/10.1093/database/baad045

Abstract

The increasing amount and complexity of clinical data require an appropriate way of storing and analyzing those data. Traditional approaches use a tabular structure (relational databases) for storing data and thereby complicate storing and retrieving interlinked data from the clinical domain. Graph databases provide a great solution for this by storing data in a graph as nodes (vertices) that are connected by edges (links). The underlying graph structure can be used for the subsequent data analysis (graph learning). Graph learning consists of two parts: graph representation learning and graph analytics. Graph representation learning aims to reduce high-dimensional input graphs to low-dimensional representations. Then, graph analytics uses the obtained representations for analytical tasks like visualization, classification, link prediction and clustering which can be used to solve domain-specific problems. In this survey, we review current state-of-the-art graph database management systems, graph learning algorithms and a variety of graph applications in the clinical domain. Furthermore, we provide a comprehensive use case for a clearer understanding of complex graph learning algorithms.

Graphical abstract



Received 9 December 2022; Revised 26 May 2023; Accepted 16 June 2023

Key points

- Review current graph databases and graph learning algorithms.
- Review graph learning applications in medicine.
- Clearer understanding by introducing a medical use case for applying graph learning algorithms.

Introduction

The amount and the complexity of clinical data is continuously increasing (1). Ongoing digitalization in the biomedical domain makes big clinical data sets more available for storage and subsequent analysis. The insights hidden in this data offer the potential for improved treatments of patients, e.g., by more accurate and faster diagnoses, examination of adverse effects and discovering of new drugs for specific targets (2). However, the rapidly increasing data present serious problems in storing and processing these data appropriately (3). Traditionally, relational databases store such data in a tabular structure and use SQL (Structured Query Language) for querying them. Recently, non-relational stores, so-called NoSQL databases, became popular to handle the shortcomings of relational databases for storing and querying big data (e.g., less flexible data schemas). NoSQL databases include key-value stores, wide-column stores, document stores and graph stores (2). Graph databases use graph stores and provide efficient entity traversals, i.e., they are ideal for handling interconnected (i.e., entities with one or multiple relationships/interactions between each other) and heterogeneous (i.e., different kind of entities) data, such as clinical data (e.g., patients with similar laboratory results having similar treatments in a graph with connections based on the laboratory results and treatments of each patient (Figure 1)) (3). Graph stores represent data in form of graphs consisting of nodes (also called vertices) and edges (also called links or relations), which connect nodes with each other (4). Nodes, edges, or complete graphs can have labels and features (also called attributes) (5). For example, a graph containing patients and symptoms might use node labels to distinguish between these two types of nodes (Figure 2). Additionally, a patient might have further information, e.g., sex, age and pre-existing conditions, stored as node features.

A large graph that represents one fact as a triplet consisting of a head-entity (node), a relation (edge) and a tail-entity (node) is called a knowledge graph (6). While we would like to stick to this definition of Knowledge Graphs for simplicity, there exist some further definitions of knowledge graphs (7). Knowledge graphs have several different applications, including Health Knowledge Graphs (8–10), Biological

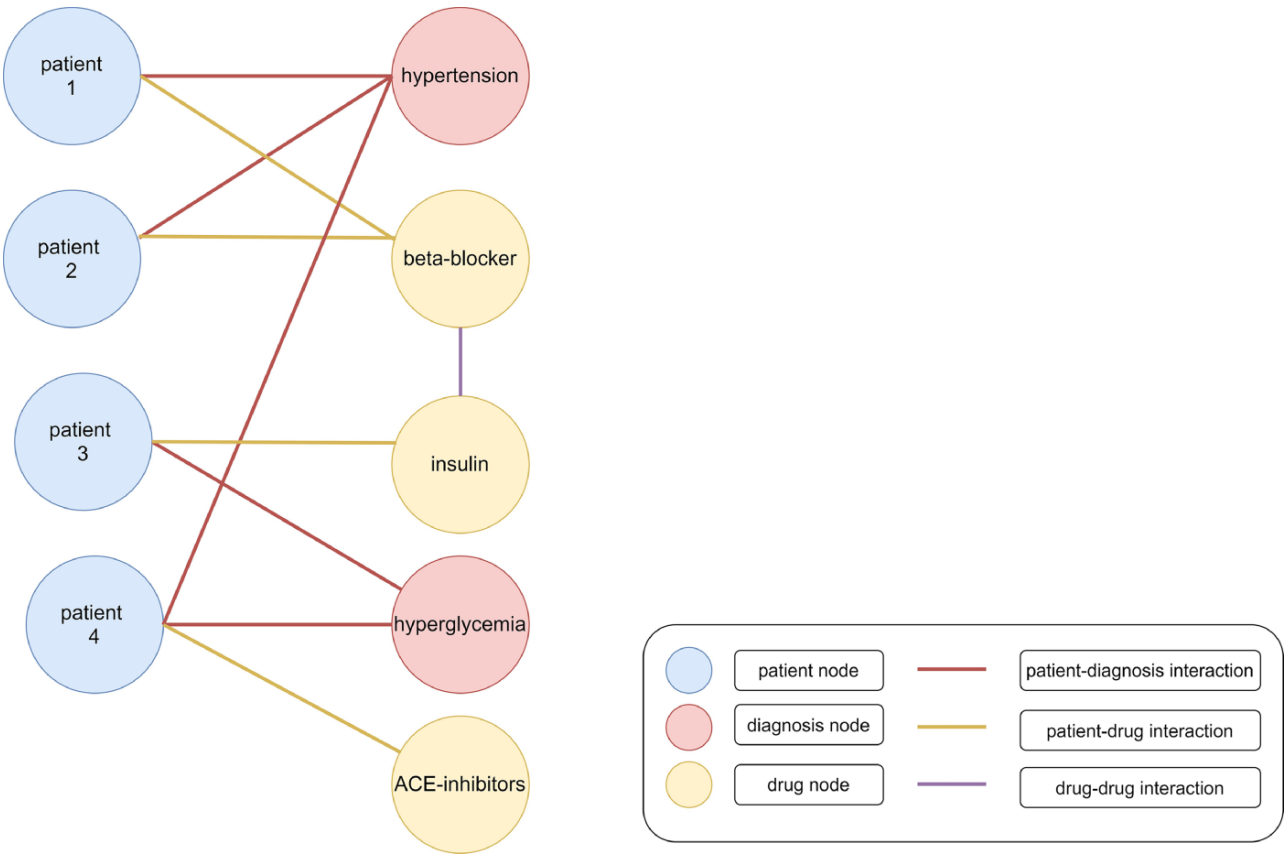


Figure 1. Heterogenous graph with clinical data. Patients (blue nodes) are connected with diagnoses (red nodes) with a red edge and drugs (yellow nodes) with a yellow edge. Interactions between different drugs are represented as a purple edge. Patients 1 and 2 were diagnosed with hypertension. Therefore, beta-blockers were administered as treatment. Patient 3 was diagnosed with hyperglycemia and cured with insulin. However, patient 4 was diagnosed with hypertension and hyperglycemia (199). The administration of beta-blockers and insulin would lead to negative side effects because of an increased risk for hypoglycemia. Therefore, angiotensin-converting enzyme (ACE) inhibitors might be administered as treatment (200).

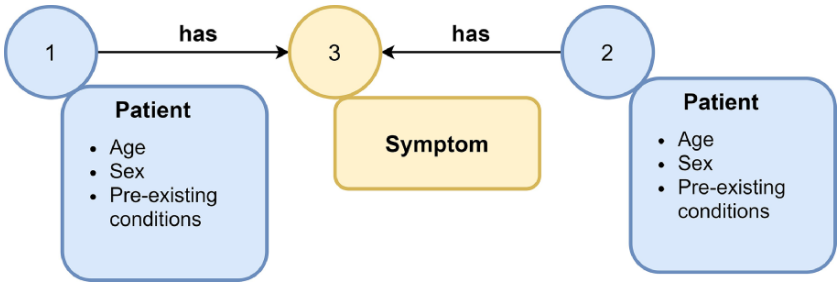


Figure 2. Example of a graph. A graph consists of nodes (represented as circles) and edges (represented as arrows) that connect nodes with each other. Nodes and edges can have labels, e.g., symptom or patient, and additional attributes. Attributes (often referred to as features) contain additional information about nodes or edges, e.g., the age and sex of a patient node.

Knowledge Graph (11, 12), Knowledge Graphs for Covid-Research (13–15) and many others (16–18). Graph databases are already used in several other applications, e.g., social networks (19–22), recommendation systems (23, 24) and fraud detection (25, 26). Several different graph database management systems (DBMSs) have been proposed, e.g., Neo4J (27), NebulaGraph (28), TigerGraph (29), DGraph (30), ArangoDB (31) and many more (32). They are distinguished from each other in their query language, their data storage model [i.e., labeled property graph (LPG) or resource description framework (RDF)], the availability of access controls, the supported programming languages and their license. One of the currently most popular graph databases, Neo4j, uses native graph storage and processing. The query language Cypher is used to query data and Neo4j offers different graph analytics algorithms in their Graph Data Science Library (27).

After choosing a graph DBMS for the desired use case, users need to design a graph data model to store their data in the graph database. While constructing an appropriate data model, users typically face the following challenges:

- Which data should be stored as separate nodes, edges, or graphs?
- Which data should be stored as node/link/graph attributes?
- What labels should the stored nodes/edges/graphs have?
- How to define attributes to make them accessible for further processing steps?

Unfortunately, there is no 'one size fits all' solution for these questions. It depends on the data, i.e., whether the data is numeric (discrete/continuous) or categorical (nominal/ordinal), and the intended use case, i.e., what questions should be answered with the data. If the goal is to predict interactions between different drugs, we would model drugs as nodes and known interactions as edges (drug–drug interaction graph) (33–35). However, if the goal is to predict properties of different drugs, we might represent each drug agent in its chemical structure as an individual graph with nodes as atoms and atomic bonding as edges (36–40).

After storing the data in an appropriate data model, the next goal is to analyze the graph data. This is where graph learning becomes important. Graph learning is the application of machine learning techniques on graph data, i.e., it simplifies complex interconnected data to draw conclusions and answer specific questions (41). It is distinguished from other machine learning techniques by making use of the graph structure (nodes and the relations between them). Graph learning consists of two parts, graph representation learning and graph analytics. Graph representation learning is used to reduce the dimensionality of the input graph, which might include millions or even billions of nodes and edges (42). There are several graph representation learning techniques, which can be divided into (i) matrix factorization-based approaches, (ii) random walk-based approaches and (iii) graph neural networks (GNNs). The output of graph representation learning is a set of low-dimensional embedding vectors (i.e., one for each node), or a single embedding (i.e., for the entire graph) that can then be used in graph analytics for graph mining tasks, or visualization tasks. The most commonly used graph mining tasks for analytics are node classification, link prediction and graph classification (42). These tasks are already used in several clinical real-world use cases, including predicting interactions between different drugs (43) and diagnosing patients based on their medical history (44, 45). Graph learning is not only a promising technology for decision-support systems and drug discovery, but also for generating new knowledge through Knowledge Graph completion (46, 47). Furthermore, modern hardware like GPUs and FPGAs can further accelerate analyzing large amounts of graph data by parallelizing some calculations (48, 49).

Organization

In this paper, we will define and explain graph databases and review several different graph databases (Section 2). Then, we illustrate the concept of graph learning, review the most important graph learning algorithms, and explain their limitations (Section 3). Finally, we show some already existing applications for graph learning in the clinical domain (Section 4.1–Section 4.4).

Our contributions

There already exist some review articles about graph databases and graph learning comparing many different approaches and embedding methods (32, 42, 50–53). Besides them, there are few publications about either graph databases in the biomedical domain or graph representation learning in bioinformatics (51, 54, 55). To the best of our knowledge, this is the first survey incorporating a complete pipeline from data storage, over graph learning to real-world applications in clinical domains (Figure 3). Especially for beginners in this field, our work helps in getting started with graph databases

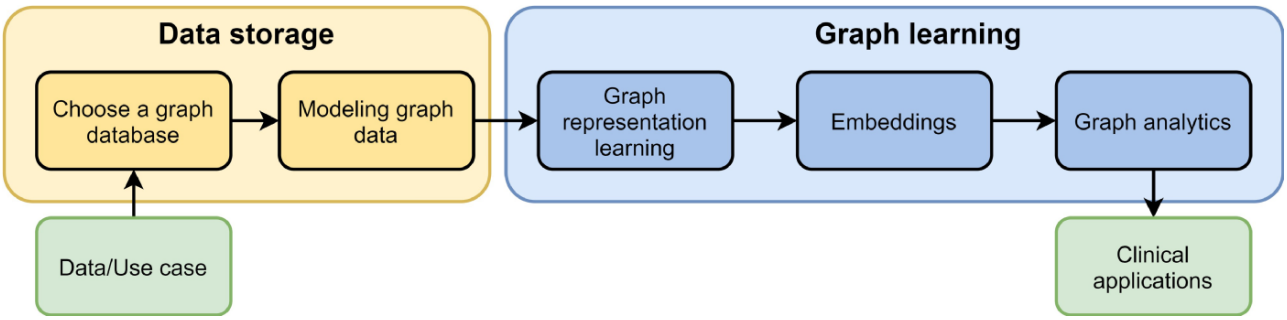


Figure 3. Pipeline for processing graph data. The pipeline consists of two parts, data storage and graph learning. As input users have data and a specific use case. After they have chosen a graph database, they need to design a graph data model to store their data appropriately (data storage). In the second part, high-dimensional graph data are reduced to obtain low-dimensional embeddings using graph representation learning. These embeddings are used in graph analytics for several different clinical applications, e.g., predicting interactions between drugs.

and provides a comprehensive introduction for processing big clinical data. While this review primarily focuses on clinical applications, the concepts introduced in this work can also be applied to many other applications, e.g., recommendation systems and fraud detection.

Graph databases

In large-scale machine learning applications, data storage is as important as the machine learning algorithms that learn patterns in the data (56, 57). Data must be securely stored, especially for clinical applications, readily available for training of machine learning algorithms and remain updateable with new information. Databases are organized as collections of data that, depending on their DBMS support different types of data and allow for access patterns that are beneficial in different applications. Historically, relational database systems have been used to store the majority of data in production settings. In this section, the journey from classic relational databases to graph databases will be explored. First, the traditional relational database management systems will be introduced, as well as the challenges that arise from joins in relational databases (Section 2.1). Secondly, it will be shown how graph databases represent data and how the graph data representation can partially overcome the challenges of relational databases (58) (Section 2.2 and Section 2.3). Finally, we will explain why we need graph query languages and name the most important representatives (Section 2.4).

Relational Database Management System (RDBMS)

In relational database management systems, single records are represented as rows in a defined table structure. The table structure is based on columns that represent the attributes of the data. Each record can be identified by a primary key. A relation can be represented by using the identifiers of records as foreign keys. These keys can be used in join tables to identify a row in one table with a row in another table.

Primary keys are a column or set of columns that uniquely identify a row in a table. Foreign keys are a column or set of columns that identify a row in another table.

SQL is the standard language for accessing and manipulating records in an RDBMS Most relational databases support SQL and it is the standard for data manipulation in RDBMS (58, 59).

Joins

Joins are query operations on a relational database that allow for the retrieval of multiple records from different tables according to a join condition. Joins therefore link these records from the RDBMS together based on the join condition. Join conditions are logical comparisons of fields or keys between tables. Joins can therefore be used for combined retrieval of related entries from two or more tables in the RDBMS. This makes Joins the preferred method for querying data based on a relationship between the elements in the table that is encapsulated in the logical condition. Joins can be separated based on the way the data is retrieved from the joined tables.

There are two types of joins defined in SQL: inner joins and outer joins. Inner joins return rows from both tables that match the join condition. Outer joins return rows from the one table that does not match the join condition (Figure 4). Therefore, left outer joins return all rows from the left table and all rows from the right table that satisfy the join condition. Conversely, right outer joins return all rows from the right table and all elements from the left table that satisfy the join condition. Full outer joins return all rows of both sides pivoted on the field in the join condition (60).

References by foreign key can be resolved by the SQL JOIN operation. Many-to-many relations in relational DBMSs require multiple JOIN operations. Relational DBMSs struggle to integrate new ad-hoc relations between records in the table structure. New relations that are introduced into the existing relational database require join tables that limit the performance of the database. However, relational databases provide materialized database views that store results from SQL operations like JOINS in a new materialized table (61). After the creation of this materialized table, one can directly query these data without requiring any JOIN operations. Thereby, views facilitate querying connected data from multiple different tables and increase the query performance. Furthermore, specific columns in this new table can be indexed to further accelerate potentially computational expensive search, comparison and filtering operations (62). Another direction to overcome performance issues in RDBMS is to use specific data structures tailored to the desired use case. One example

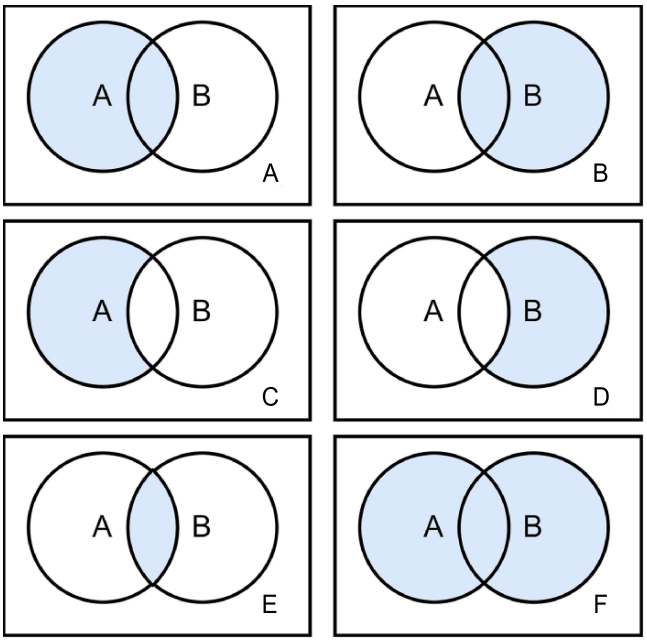


Figure 4. Join Types. There are six different join types: left Outer Join (A), right Outer Join (B), left Outer Join with Null (C), right Outer Join with Null (D), inner Join (E) and full Outer Join (F). Each circle represents a table and the blue color highlights the retrieved data from the joined tables.

might be the integration of more information in a single table (e.g., denormalization to WideTables (63)) to prevent potentially expensive JOIN operations.

Graph Database Management System (GDBMS)

NoSQL databases are non-relational databases. Contrary to relational databases, these databases do not represent their data in a table structure. The field of NoSQL databases includes a variety of databases with different data models; these include but are not limited to document-oriented databases, key-value databases, wide-column databases, RDF stores and native graph databases (2, 58).

Graph database management systems (GDBMS) are NoSQL databases that use graph structures for semantic queries with nodes, edges and properties to represent and store data and the relationships exhibited by the data. A key concept of the system is the graph, which can be modeled in various ways. The direct relationships of data in the graph allow data in the store to be linked together directly and, in many cases, retrieved with one operation. These retrievals can be powerful when the data is interconnected. A graph database can be used to represent any kind of data that has relationships between data elements. Therefore, graph databases are often used in applications where the underlying data has a lot of relationships between items, or where the relationships between data items are more important than the individual items.

Native graph databases

A database that models graph data can be implemented in any NoSQL data model and even relational databases (32). Different data representations therefore require different concepts to represent nodes, edges, labels and properties contained in a graph. Graph databases exist that are built on tuple stores, wide-column databases, key-value databases, document databases and relational databases.

However, the LPG and RDF are two data models that implement a graph natively for utilization as a database.

Differences between relational and graph databases

Compared to graph databases, relational databases perform better in respect to query efficiency and data modeling for storing data that can be easily normalized into a tabular format (58). Graph databases are better suited for storing and accessing data where most data is interconnected, such as social data, location data, network data and biomedical data (Figure 5). Relational databases are better for online transaction processing (OLTP) applications because relational databases are better at supporting transactions and maintaining data integrity. Relational databases are typically more expensive to maintain and to scale than graph databases because inserting new data elements with relationships to old data elements requires join operations for the integration into the existing tabular data model. Many different data models can be used to implement graph databases. The most important native Graph data models are RDF stores (64) and LPG (32).

Graph data models

Graph model

A graph G(V, E) is composed of an ordered pair of two disjoint sets: vertices V (also referred to as nodes) and edges (or links) E (32). In information theory, a graph is a representation of a set of objects and relationships between them. Graphs can be used to model a wide variety of structures, including networks, hierarchies, ontologies and other forms of interconnected data. In native graph databases, a graph can be represented as either an LPG or as RDF triples (64).

Labeled Property Graph (LPG)

Property graphs use a node-edge-property model, where each node represents an entity, each edge represents a relationship between two entities, and each property represents an attribute of an entity or relationship (Figure 6). The property graph model is based on the classical graph model, but introduces labels to vertices, edges, and properties. Labeled vertices and edges can represent different classes of data. Properties augment the vertices and edges with key-value pairs, where the key identifies the property and the value represents the feature of the property (32, 65).

Neo4j

Neo4j is a GDBMS that uses native graph storage and processing. Neo4j uses the LPG data model and the Cypher query language. Neo4j graphs are directed however this limitation can be overcome at query time (27, 32, 66).

SparkSee

Sparksee is an LPG graph database management system. Sparksee allows directed and undirected graphs. Sparksee uses the OpenCypher query language (32, 67).

TigerGraph

TigerGraph is a property graph database, which supports the LPG model. TigerGraph uses a SQL-like query language

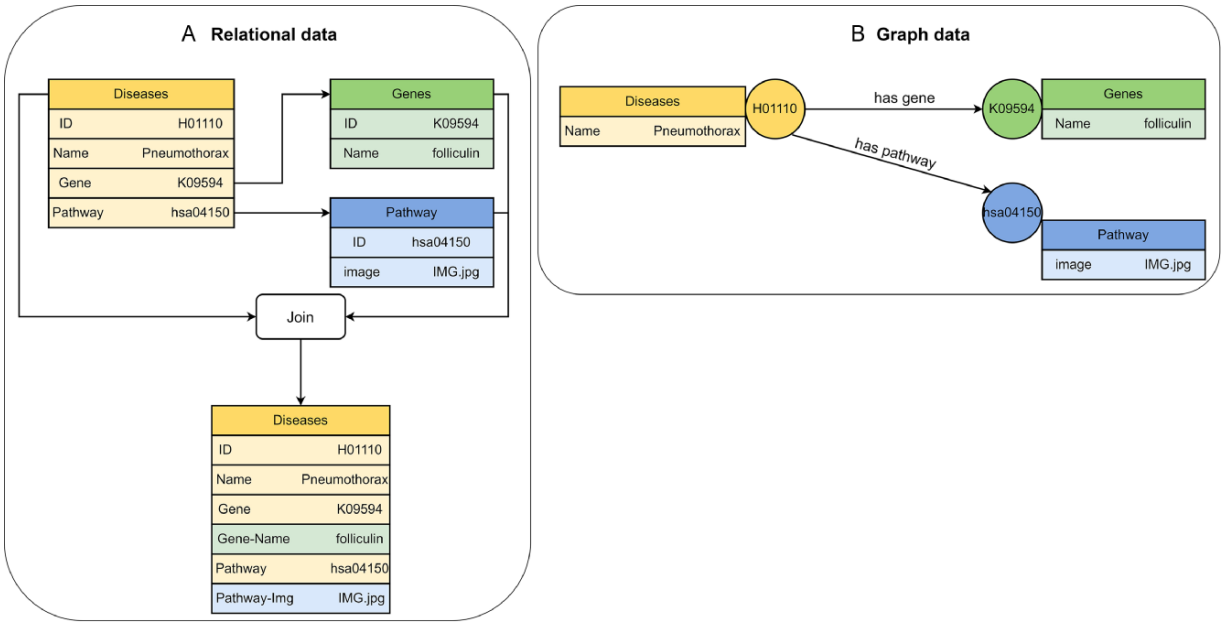


Figure 5. Visualization of (A) relational data and (B) graph data of disease, gene and pathway entities in KEGG (201). A disease references to a pathway entity and to a gene entity. For receiving information from multiple tables in a relational database, we must perform join operations on the tables. In a graph database, we traverse edges to receive information from multiple nodes.

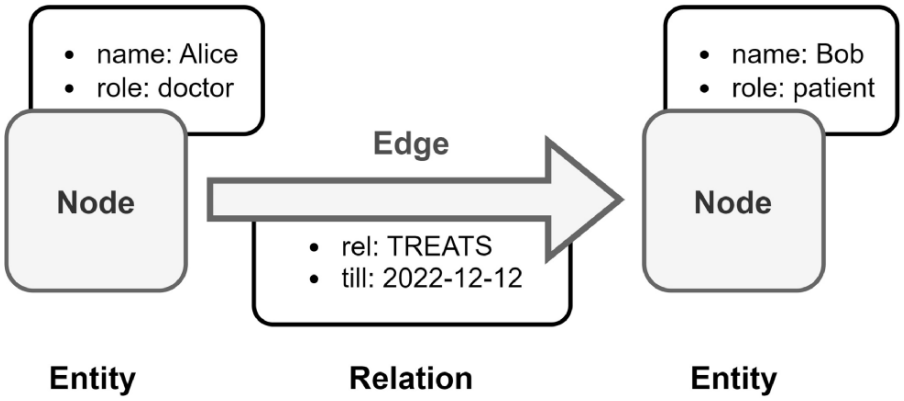


Figure 6. Overview of Labeled Property Graph (LPG). In this example, medical data is transformed into an LPG format. Alice is a doctor and Bob is a patient, both are represented as nodes in the LPG. The doctor–patient relationship between Alice and Bob is modeled by a directed edge in the LPG with an assigned property, which is the end date of Bob's treatment by Alice.

(GSQL). TigerGraph offers built-in MapReduce operations and parallelism (29, 32).

Resource Description Framework (RDF) stores

The RDF (68) was first published by the World Wide Web Consortium (W3C) in 1997 as a collection of specifications for the representation of information. The RDF was revised in 2014 to version 1.1. The goal of the RDF is the standardization and simplification of the exchange of ontologies. Ontologies are sets of terms and the relationships between them. To facilitate this goal, RDF triples use a subject-predicate-object graph model, where each triple represents a relationship between two items in the graph (Figure 7). Subjects are Uniform Resource Identifiers (URIs) or blank nodes, objects can be URIs, literals or blank nodes and predicates are URIs. The collection of these information triples forms a collection that is often referred to as a triple store. RDF triples are a standard format for linked data (32, 69).

Graph query languages

Besides their data model, databases can be classified by their query language. A query language is used for the retrieval and access to data in databases. These query languages facilitate the access, manipulation, and creation of data in the database. In graph databases, the data model also influences the used query language. While most RDF databases rely on the SPARQL Protocol and RDF Query Language (SPARQL) for data access and manipulation, most LPG databases employ Cypher, Gremlin, or GSQL as their query language (70). Tables 1–3 provide an overview how to create a graph (Table 1), create/delete nodes (Table 2), and create/delete edges (Table 3) in different query languages.

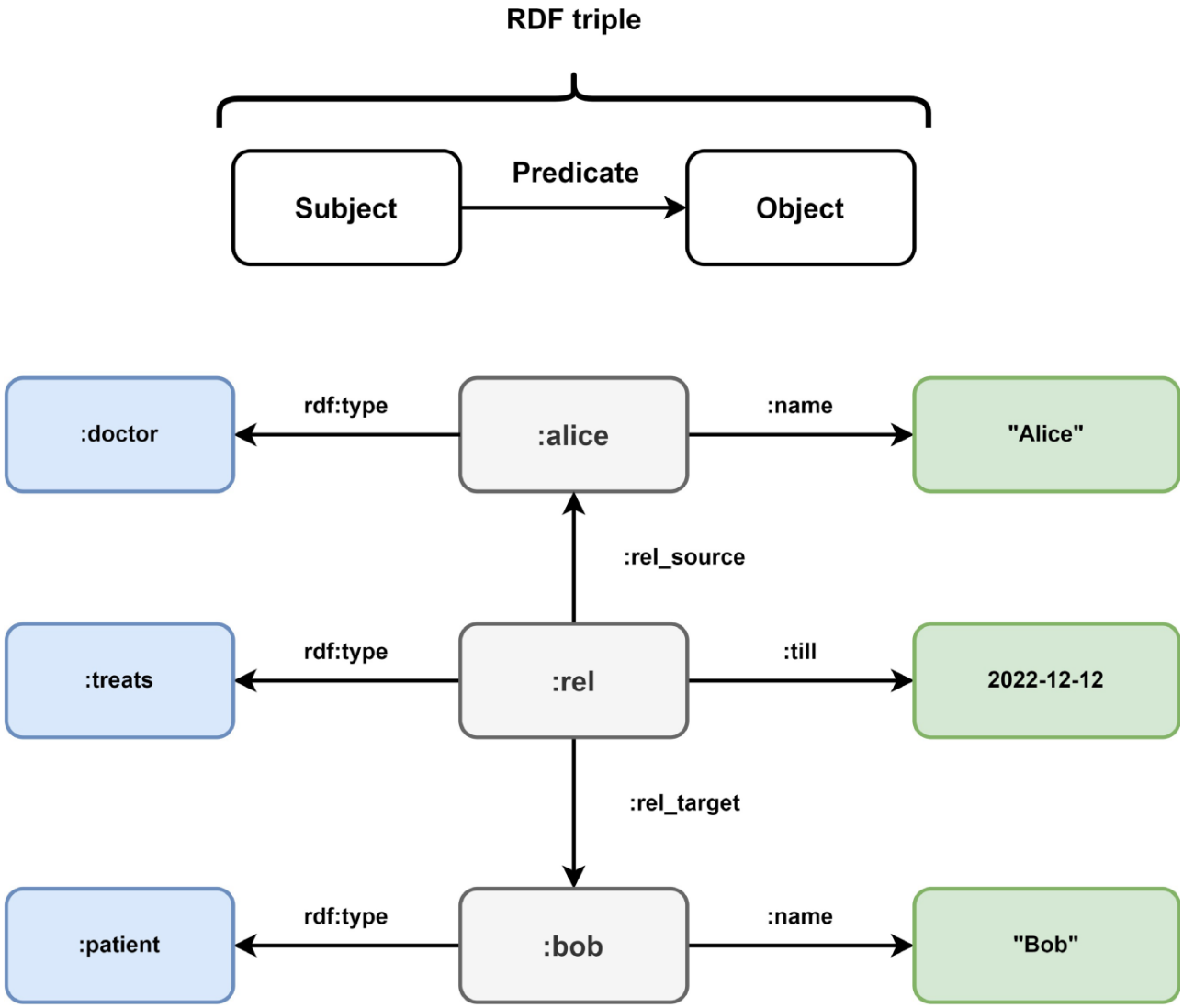


Figure 7. Overview of Resource Description Framework (RDF). Every information in the Resource Description Framework is organized as a triple and every triple follows a subject predicate object structure. In this example, the medical data of Figure 5 is transformed into the RDF format. The subject:alice has the predicate rdf:type with the object:doctor and the predicate:name with object 'Alice'. These RDF triples represent the node of doctor Alice in the LPG. The directed nature of the edge TREATS in the LPG is represented by the RDF triples (:rel,:rel_source,:alice) and (:rel,:rel_target,:bob). The properties of the edge are represented by the triples (:rel, rdf:type,:treats) and (:rel,:till, 2022–12-12). Bob is represented by the triples (:bob, rdf:type,:patient) and (:bob,:name, "Bob).

Cypher

Cypher is Neo4j's own declarative query language (71). Its main building blocks are 'patterns'. These patterns are constructed as nodes in squared brackets and connecting edges as arrows (65).

GSQL

GSQL is TigerGraph's high-level graph querying and update language with a declarative semantic. It is closely modeled after SQL from relational databases (SELECT-FROM-WHERE as the core building block). Therefore, GSQL is compatible with SQL (72, 73).

Gremlin

Gremlin is a graph traversal language introduced in 2009. Gremlin is maintained and developed by Apache TinkerPop (74). Gremlin supports imperative and declarative querying of graph data in graph databases. Apache TinkerPop is a graph computing framework for both OLTP and online analytical processing applications (65, 74).

SPARQL

SPARQL is a semantic query language introduced in 2008. SPARQL is able to retrieve, manipulate and store data in the RDF format (65, 75).

Graph learning

After loading all data in the chosen graph database in an appropriate data model, data analysis becomes important. Graphs are non-Euclidean data structures (76), i.e., nodes in a graph can have a complex topological structure and do not have a natural order like pixels in a picture or words

Table 1. Overview of how to create graphs using different graph query languages

Query language	Create graph
Cypher	CREATE DATABASE graphName
GSQL	CREATE VERTEX nodeType (PRIMARY_ID idName idType, feature1 featureType) CREATE UNDIRECTED EDGE edgeType (FROM nodeType, TO nodeType, featureValue featureType) CREATE GRAPH graphName (nodeType, edgeType) USE GRAPH graphName
Gremlin	graphName = TinkerGraph.open().traversal()
SPARQL Update	CREATE GRAPH IRIref

Table 2. Overview of how to create and delete nodes using different graph query languages

Query language	Insert nodes	Delete nodes
Cypher	CREATE (n:nodeType {feature1: value})	MATCH (n:nodeType {feature1: value}) DELETE n

GSQL	INSERT_INTO nodeType VALUES (idValue, featureValue)	CREATE QUERY queryName() FOR GRAPH graphName {
		N = {nodeType.*};
		DELETE n FROM N:n
		WHERE n.feature1 == featureValue;
		}
		RUN QUERY queryName()

Gremlin	graphName.addV().property(id, idValue).property(feature1, value).next()	graphName.V().hasId(idValue).drop()
SPARQL Update	INSERT DATA {GRAPH IRIref {URI_subject URI_predicate value}}	DELETE DATA {GRAPH IRIref {URI_subject URI_predicate value}}

Table 3. Overview of how to create and delete edges using different graph query languages

Query language	Insert edges	Delete edges
Cypher	MATCH (m:nodeType), (n:nodeType) WHERE m.feature1 = featureValue AND n.feature1= featureValue CREATE (m)-[e:edgeType]->(n)	MATCH (m:nodeType)-[e:edgeType]-> (n:nodeType) DELETE e

GSQL	INSERT INTO edgeType (FROM, TO, feature1) VALUES (startNode nodeType, endNode nodeType, featureValue)	CREATE QUERY queryName() FOR GRAPH graphName {
		N = {nodeType.*};
		DELETE e FROM N:n -(edgeType:e)- nodeType:m
		WHERE n.feature1 == featureValue;
		}
		RUN QUERY queryName()

Gremlin	graphName.addE('edgeLabel').from(startNode).to(endNode).property(id, idValue).property(feature1, value)	graphName.E().hasId(idValue).drop()

in a sentence. In this work, a topologically complex structured graph is defined as a graph with a large number of nodes (thousands, millions or even billions of nodes) which are connected by many edges based on one or multiple relationships. Several traditional approaches (Section 3.1) were proposed within the last decades to analyze graphs with different measures, statistics, or kernels (41, 77–84). They represent the pioneer works for analyzing graph data, but they are transductive (i.e., they can only learn from observed nodes) and computationally inefficient. Recently, machine learning on graphs (graph learning) has become popular. It consists of two parts, graph representation learning (Section 3.2) and graph analytics (Section 3.3). Graph representation learning transfers a given topologically complex structured input graph into a more accessible format. In graph analytics, graph learning uses this format for several tasks including visualization, classification, and prediction tasks. Table 4 lists advantages and disadvantages of traditional approaches, matrix-factorization-based methods, random-walk-based methods, and GNNs.

Traditional approaches (manual feature extraction)

Traditional approaches can be divided into approaches for node-level, graph-level, and edge-level feature extraction. Machine learning classifiers (e.g., logistic regression) use the resulting features for making predictions. Node-level features can encode information like the number of neighbors of each node (node degree), a node's importance in a graph (node centrality), and a node's neighborhood cluster density (clustering coefficient) (85). Graph-level features can extract information by aggregating statistics/features from

Table 4. Advantages and disadvantages of different categories of graph learning approaches used for analyzing graphs

Category	Advantages	Disadvantages
Traditional approaches (manual feature extraction)	Easy to use and understand	Require careful, hand-engineered statistics/measures Time-consuming and expensive design (35)
Matrix-factorization-based	Capture global structure (44)	Cannot use features, no parameter sharing, transductive, deterministic measure of neighborhood overlap (35, 36), high time and memory cost (44)
Random-walk-based	Stochastic measures of neighborhood overlap (35)	Cannot use features, no parameter sharing, transductive (35)
Graph neural networks	Most can use features, Most are performant, Can process large and complex data, Some are inductive (35, 73)	Low interpretability, risk of over-smoothing, under-reaching and over-squashing (35, 36, 74,76)

all nodes within a graph, iterative neighborhood aggregation (Weisfeiler-Lehman kernel), or counting the occurrence of different small subgraph structures (graphlet kernel) (77–81, 85). Finally, edge-level feature approaches extract features by counting the number of neighbors that two nodes share (local neighborhood overlap detection) or counting the numbers of paths of all length between two nodes (global neighborhood overlap detection) (41).

Graph representation learning

Traditional approaches require careful, hand-engineered statistics and measures, and their design is time-consuming and expensive (42). Graph representation learning provides a more flexible approach and aims to find efficient task-independent representations of nodes in a graph or subgraph. Therefore, these algorithms aggregate information of each individual node as well as information from its local neighborhood into a single vector (embedding). Depending on the task (node-level, edge-level, or graph-level task), these embeddings are either directly (node-level tasks) passed to state-of-the art machine learning (SOTA-ML) algorithms or the embeddings are further aggregated (edge-level task or graph-level task) and then passed to SOTA-ML algorithms for final predictions (graph analytics, Figure 8). Besides these so-called graph mining tasks, the resulting embeddings can also be used for visualizing data (Section 3.3). Modern graph representation learning can be divided into three broad categories—matrix factorization-based methods, random walk-based methods and GNNs.

Matrix-factorization-based methods

Matrix-factorization-based methods represent node connections in the form of a similarity matrix (e.g., adjacency matrix, Laplacian matrix, k-step transition probability matrix) and use matrix factorization for generating embeddings (50, 53). Intuitively, they aim to learn embeddings so that the multiplication of node embedding i and j approximates their similarity (42). The factorization-based approaches vary based on the matrix properties (53). The most common representatives of matrix-factorization-based approaches include locally linear embeddings (LLE) (86), Laplacian Eigenmaps (87), Graph Factorization (88), HOPE (89) and GraRep (90, 91).

Random-Walk-based methods

Random-walk-based approaches learn embeddings based on random walk statistics (42). Random walks are very powerful shown in several applications like Google's page rank algorithm (92). A random walk collects a series of nodes based on their connection (Figure 9A). It starts from a specific node and picks the next node randomly from the node's neighborhood. The latter step is repeated n times, where n represents the length of the random walk. As a result, we receive for each node a set of node sequences which represent the neighborhood of the node. Finally, we use these neighborhoods in an optimization function to find node embeddings (i.e., vectors) so that nodes on the same sequence are embedded close to each other (i.e., have similar vectors) (Figure 9B). Thereby, random-walk-based approaches allow measuring the similarity of a node based on its sampled neighborhood (42). Random-walk-based approaches include techniques like DeepWalk (93) and node2vec (94). DeepWalk uses successfully established deep learning techniques of natural language processing for graph analysis. Short random walks generate latent representations of nodes in a graph to preserve higher-order proximity between nodes. Obtained random walks are then used to train a machine learning model for receiving the final embeddings (93). Node2vec is an algorithmic framework for generating node embeddings, while making a trade-off between breadth-first sampling and depth-first sampling (94). Furthermore, Chen *et al.* proposed HARP, a meta-strategy for improving random-walk based methods like DeepWalk and Node2vec (95).

Graph neural networks

Matrix factorization-based methods and random walk-based methods have two serious limitations. First, they cannot generate embeddings for unseen nodes (transductive learning), i.e., a complete re-computation is needed as soon as new nodes appear in the graph. Second, they cannot use features attached to nodes, edges, or graphs. GNNs provide (at least partially) solutions for these limitations (42). They can be divided into graph autoencoders, recurrent GNNs, convolutional GNNs, and spatial-temporal GNNs. Graph autoencoders consist of two parts: an encoder and a decoder. While the encoder reduces the dimensionality of the high-dimensional input, the decoder tries to reconstruct the graph from the embedding. Examples of the neighborhood autoencoders

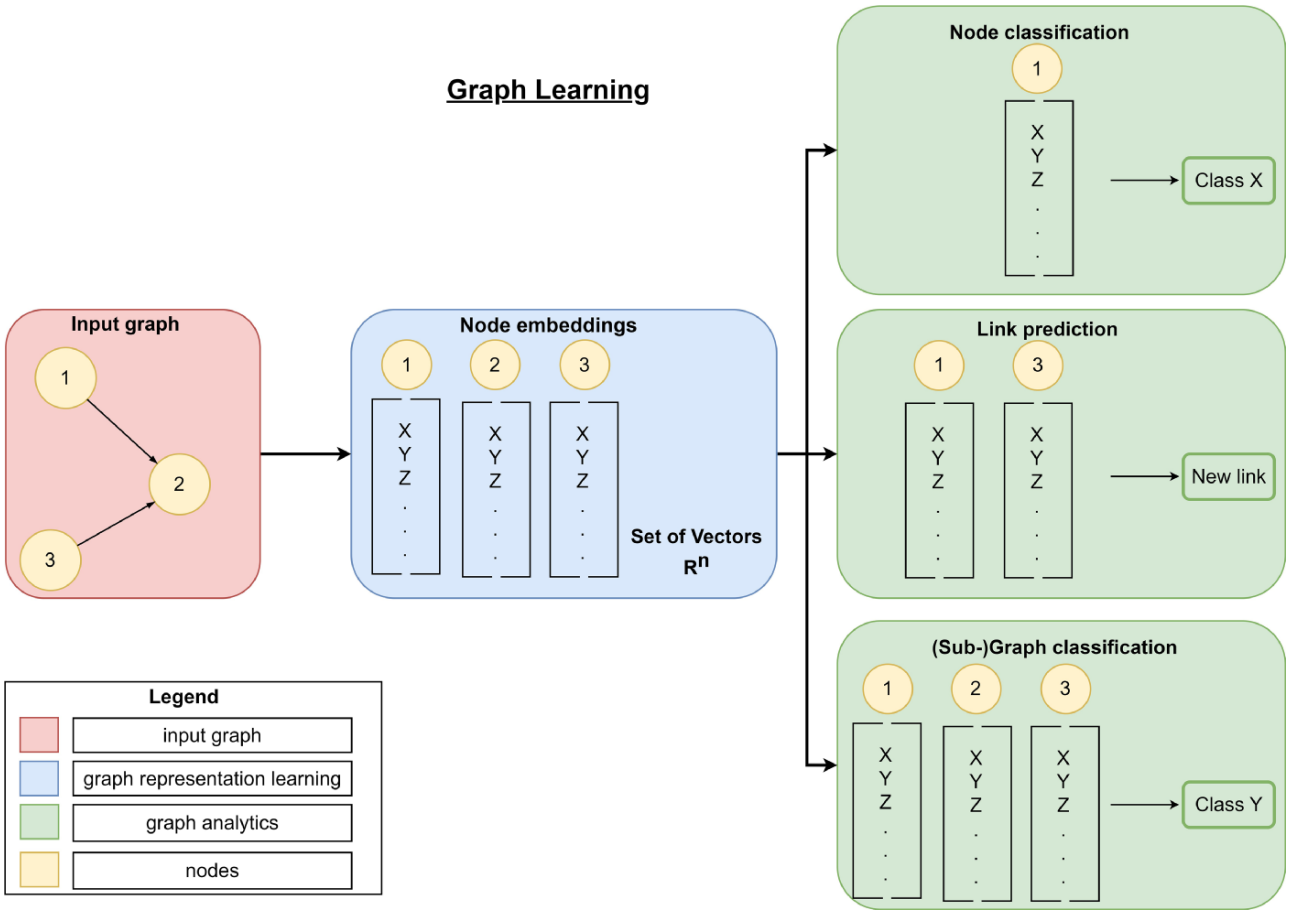


Figure 8. Overview of graph learning (machine learning on graphs). Graph learning can process a high-dimensional input graph (red rectangle) using graph representation learning (blue rectangles) and graph analytics (green rectangles). Graph representation learning allows the generation of low-dimensional node embeddings using graph representation approaches (manual feature extraction, matrix-factorization-based, random-walk-based, or graph neural networks). These embeddings represent feature information and topological information about a node efficiently. Then, graph analytics use these low-dimensional embeddings for graph mining tasks like link prediction to predict new links, node classification to classify nodes or (sub-)graph classification to classify entire (sub-)graphs.

are Deep Neural Graph Representations (DNGR) (96) and Structural Deep Network Embeddings (SDNE) (97). While graph autoencoders can incorporate structural information about a node's local neighborhood, they are still transductive like random walk-based and matrix-factorization-based methods, and they are computationally expensive due to their fixed input dimension to the autoencoder (42). Recurrent GNNs represent the pioneer works of GNNs and learn node representations with recurrent neural architectures (52). Information between nodes is constantly exchanged until a stable equilibrium is reached. Gated Graph sequence neural networks (98) and graph echo state networks (99) are representatives of recurrent GNNs. The idea of information exchange between nodes motivated recently proposed spatial convolutional GNNs. Convolutional GNNs can be divided into spectral and spatial approaches and provide solutions for the previously mentioned limitations of other graph representation learning techniques. Spectral approaches introduce filters from the perspective of graph signal processing (52). Spatial approaches are based on aggregating information from a node's local neighborhood (message passing framework) (100). In the message passing framework node features are first transformed using a learnable weight matrix and a non-linear activation function (e.g., sigmoid function) and then aggregated to obtain new node embeddings. The process of transformation and aggregation is repeated n times, where n is the number of layers. One advantage of spatial convolutional GNNs is that the underlying message passing framework allows to parallelize the operations on each node (i.e., transformation and aggregation) on modern hardware like GPUs to increase computational efficiency. Furthermore, they allow to share trainable parameters (i.e., the learnable weight matrix) between nodes by using the same weight matrix for all node features in one layer which make them statistically and computationally more efficient than spectral approaches. Several different spatial convolutional GNNs were proposed which differ in their architecture, especially their aggregation (e.g., aggregating all neighbors, random walks for aggregation) and pooling (e.g., max-pooling, sum-pooling, average-pooling) parts (100). It is noteworthy that pooling operations need to be permutation-invariant, because of the missing natural order of nodes in a graph (42). GNNs can process homogeneous graphs (i.e., graphs containing only one type of nodes and one type of edges) using homogeneous GNNs

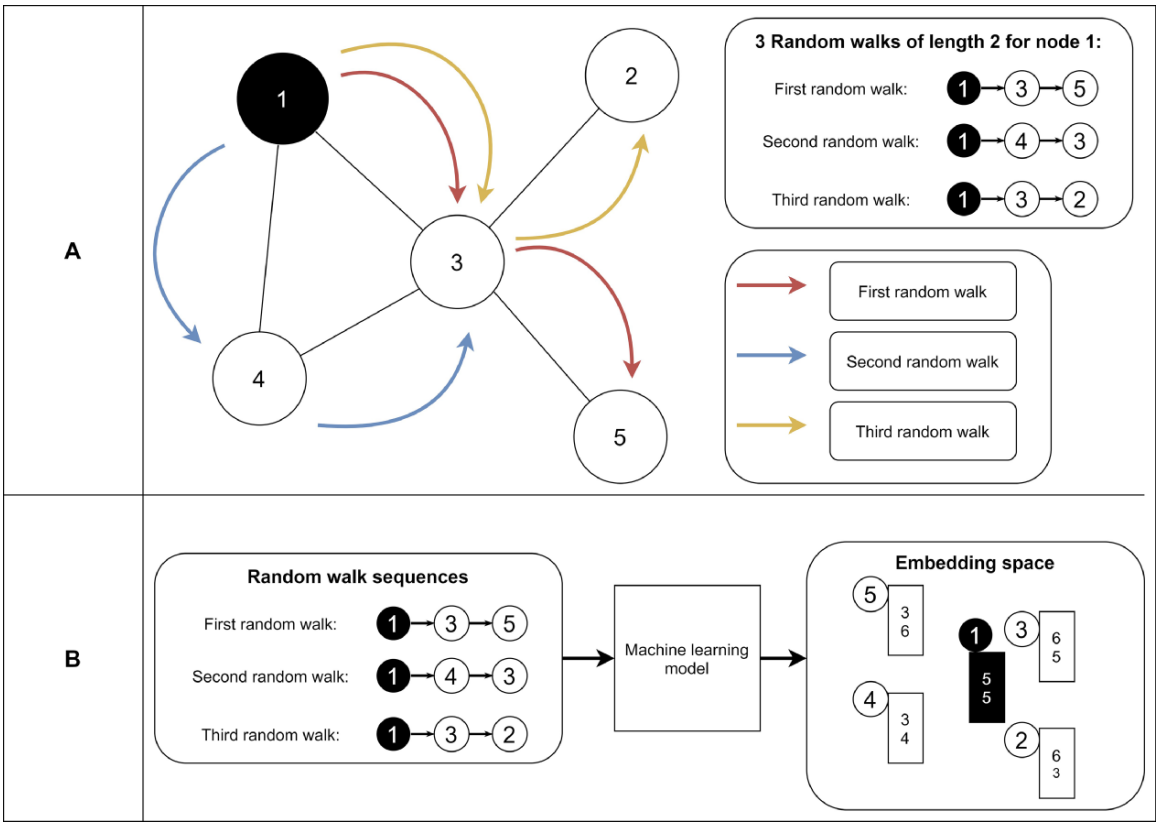


Figure 9. Example of random walks on a graph (A) and the transformation into node embeddings (B). In (A), three random walks with length two are sampled from node 1 to receive node sequences as a result. These node sequences represent the neighborhood of node 1. Then, we aim to find node embeddings such that nodes that co-occur often on random walks are embedded closer to each other (B). Note that the sequences generated from the random walks and the numbers for representing the final node embeddings are fictitious to make the embedding generation clearer.

(98, 101–104) and heterogeneous graphs (i.e., graphs containing multiple types of nodes and/or edges) using heterogeneous GNNs (105, 106). Spatial-temporal graph neural networks (STGNNs) can consider spatial and temporal dependencies simultaneously, which becomes especially important when analyzing time series on graphs. Therefore, in most cases 1D-Convolutional Neural Networks (CNNs) are combined with graph convolutional layers to learn temporal and spatial dependencies respectively (52). The CNN assigns weights and biases to different timestamps of the time series to consider their importance.

Problems and limitations of GNNs

Although GNNs are powerful graph representation techniques, there exist some problems and bottlenecks for them. In the following, we will discuss the problem of over-smoothing, under-reaching and over-squashing in GNNs. Homophily (i.e., nodes that are connected to each other are similar) is an assumption of the message passing framework in GNNs and leads to smoothing of graphs. Smoothness means in this context that node representations become like each other. This smoothness alleviates the classification of nodes in the subsequent prediction step. Over-smoothness occurs when stacking too many layers in a GNN. Consequently, node representations become indistinguishable from each other, worsening the classification accuracy of the GNN (107). Under-reaching occurs when nodes are more than k-hops away from each other, where k is the number of layers in the used GNN, i.e., the nodes are unaware of each other. This is especially important for problems that require long-range information (large problem radius). Adding additional layers to the GNN prevents under-reaching (108). The problem is that adding further layers increases a node's receptive field (nodes that are k-hops away) exponentially and can thereby lead to over-squashing. Messages from the exponentially growing receptive field are propagated and compressed into fixed-size vectors. Thereby, the graph only learns short-range signals and fails message propagation from distant nodes (108).

Other representation learning approaches

There are also some graph representation learning algorithms which cannot be categorized in one of the previous three broad categories. They include popular methods like the large-scale information network embedding (LINE) (109) and GraphGAN (110). LINE can capture first-order and second-order proximity of each node in a graph. It is an embedding technique for handling especially large graphs due to its efficiency (50, 109). GraphGAN consists of a generator and a discriminator, which play a game-theoretical minimax-game. While the generator tries to approximate the ground-truth connectivity distribution of each node in a graph, the discriminator tries to distinguish between the connectivity from ground-truth and the generator (50, 110).

Embedding (sub-)graphs

The previously presented approaches (matrix-factorization-based, random walk-based approaches, and GNNs) can generate embeddings for each node in a graph. The obtained node embeddings can also be used to generate (sub-)graph embeddings by aggregating all node embeddings in the (sub-)graph (111). The approaches for aggregating node embeddings can differ, e.g., summing (or averaging) node embeddings (112), or introducing graph coarsening layers (113, 114). Another approach for generating (sub-)graph embeddings is by creating a virtual super-node connected to all nodes that should be included in the (sub-)graph embedding (115). The resulting (sub-)graph embedding can be used for graph-level prediction tasks in graph analytics.

Graph analytics

After graph representation learning, the obtained low-dimensional embeddings can be used for several graph analytic tasks.

Visualization

Most embedding vectors have between 16 and 128 dimensions, i.e. they cannot be visualized directly in a two-dimensional diagram. Techniques, like t-distributed stochastic neighbor embedding (t-SNE) (116) and principal component analysis (PCA) (117), can be used to reduce the dimensionality of the embeddings and thereby enable visualizing them (Figure 10). Such visualizations help to better understand the model output. It facilitates the identification of outliers or anomalies and allows to identify the margins specific embeddings differ from each other.

Graph mining tasks

Besides visualization, there are several other graph analytic tasks that we sum up to graph mining tasks, including link prediction, node classification, (sub-)graph classification, link/node/graph regression, graph generation, node clustering, and network compression. Link prediction aims to find new missing edges (relations) or remove wrong (or in the future disappearing) edges between existing nodes (118, 119). Pairs of node embeddings are used (e.g., by aggregating and linearly transforming the embeddings or by calculating their dot product) to make link predictions (42). In node classification, unlabeled nodes in a graph should be classified based on their features and their position in the graph. This is often a form of semi-supervised learning (i.e., only a subset of nodes has labels) and the goal is to predict the missing labels using the final node embeddings (41). The labeling of (sub-)graphs is accomplished by (sub-)graph classification with (sub-)graph embeddings and provided labels (41). Link/node/graph regression allows the estimation of specific numeric properties of nodes, or (sub-)graphs respectively (41, 120). Graph generation allows the generation of new graphs with specific desired properties and structures (120). Based on the structure and/or properties of nodes, a graph can have multiple different clusters that can be identified with node clustering (e.g., spectral clustering (121)). With the help of network compression, the information stored in a graph is compressed to reduce space requirements (e.g., by utilizing graph embeddings) (52).

Applications of graph learning

Several different applications of graph learning highlight the potential of utilizing a graph structure and graph learning in various clinical use cases (54, 55). In this section, we classify several clinical applications based on the following graph mining tasks:

- Link prediction (Section 4.1);
- Node classification (Section 4.2);
- (Sub)graph classification (Section 4.3);
- And further graph mining tasks (Section 4.4).

Furthermore, we want to clarify the application of graph learning on a specific use case for a clearer understanding of graph learning (Supplementary Graph learning use case).

Applications of link prediction

Link prediction aims to find missing or wrong associations between nodes and involves different applications, including modeling drug-drug interactions (43, 122), drug–protein interactions (123), protein–protein interactions (124), disease–disease interactions (125), and knowledge graph completion (46, 91, 126). Drug–drug interactions are used to predict adverse side effects when a patient takes multiple drugs simultaneously (122), and to study drug similarity for inferring novel properties of drugs for drug discovery (127). Use cases of drug-protein networks include predicting drug-target interactions for finding new therapeutic effects of drugs (123). Predicting protein-protein interactions improves our understanding of biological interactions in cells (124). In a comorbidity network (i.e., a graph where diseases are linked if the disease couple affects at least one patient), link prediction is used to predict the onset of future diseases (125). Real-world knowledge graphs widely suffer from incompleteness (91). Therefore, knowledge graph completion is an important task for finding missing relations and inferring new facts (e.g., in medical knowledge graphs). Several different models were proposed for this purpose (47, 128–133). It becomes especially important for dynamic knowledge graphs that change over time (91). Besides predicting missing links in a graph, link prediction can also be used for predicting whether links might disappear in the future (negative links), e.g., in a gene expression or medical referral network (119, 134).

Applications of node classification

Node classification aims to find labels for unlabeled nodes in a graph. This task is important for population-based datasets, i.e., one graph represents an entire community, and one node represents a single patient. Node classification is used for predicting autism spectrum disorder (135, 136), Alzheimer's disease (135–138), Parkinson's disease (137, 138), attention deficit hyperactivity disorder (139), and depression (140). Often imagery data (e.g., functional Magnetic Resonance Imaging (fMRI) data, electroencephalography (EEG) data) and non-imagery data (e.g., gender, age) are used as features attached to individual nodes. In histopathology, node classification was used for segmenting images into diagnostically relevant regions (e.g., for diagnosing prostate cancer) (141). Ma and Zhang proposed AffinityNet for disease type prediction

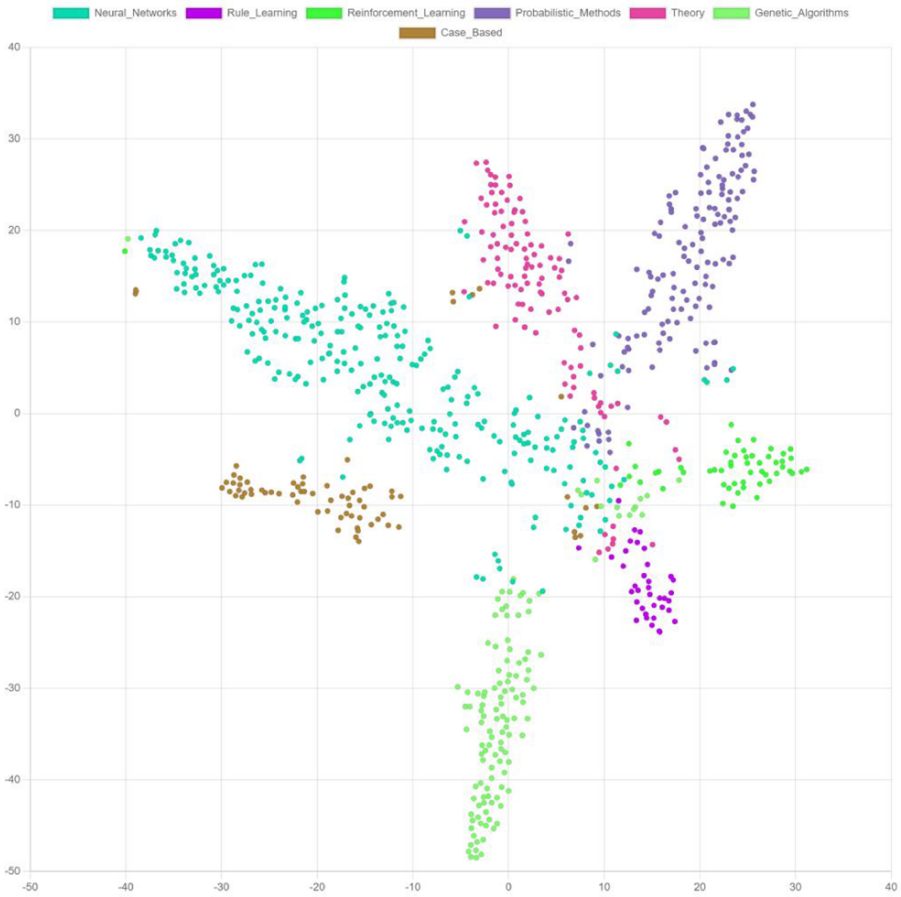


Figure 10. t-SNE. Visualization of final node embeddings after applying two graph convolutional layers on the Cora citation dataset (202) (PyTorch Geometric (203), embedding-dimension: 128). Each color represents a research topic for a node (circle).

and applied it to cancer genomic data (142). Furthermore, node classification was applied for annotating protein functions (94, 102, 143). Besides, Yue *et al.* classified semantic types of medical terms in a medical term co-occurrence graph (134).

Applications of (sub-)graph classification

Instead of representing an individual patient as a node with attached features, imagery data from fMRI, EEG, or computer tomography (CT) can be represented as an entire graph. Therefore, data is transferred into a graph structure and graph classification is applied afterwards. Several use cases highlight the applicability of this procedure using EEG data, including seizure prediction (144) and detection (145–147) in epileptic patients, identification of visual stimuli (148), emotional video classification (149), emotion recognition (150–156), and sleep stage classification (157). Furthermore, graph classification was applied to MRI and fMRI data for diagnosing Alzheimer's disease (158, 159), bipolar disorder (160), autism spectrum disorder (161), early mild cognitive impairment (162), Parkinson's disease (163), for discovering novel biomarkers (e.g., for neurological disorders) (161, 164), for subject-sex classification (159), for measuring functional connectivity between different brain regions (165), and for the identification of regions of interest in epilepsy networks (166). Histopathological images were represented as a graph to apply GNNs for the identification of lung cancer subtypes (167, 168), breast cancer (169, 170), basal cell carcinoma (171), prostate cancer (172), classification of intestinal glands (173), and predicting lymph node metastasis (174). It was also applied to CT data for COVID-19 induced pneumonia (175) and detecting COVID-19 (175, 176). Finally, graph classification was used in drug discovery for predicting molecular properties (115) and protein interfaces (177).

Applications of further graph mining tasks

Besides applications for the previously mentioned most prominent graph mining tasks, there are further ones for other graph mining tasks. For example, node regression was applied for predicting the brain age of human subjects (178). The gap between the estimated brain age and the true (chronological) age is important for understanding biological pathways relevant to aging, assessing risks for brain disorders, and developing new therapies. Another application of graph learning is learning the graph structure of Electronic Health Records to improve the performance of downstream prediction tasks, e.g., heart failure prediction (179). Graph regression was applied for predicting survival outcomes for glioma and clear cell renal carcinoma (180), and for determining colon cancer stage (181) in histopathology. Furthermore, Sureka *et al.* proposed an approach for modeling histology tissue as a graph of nuclei and used convolutional GNNs for visualization and diagnosis of diseases like breast and prostate cancer (182).

Another graph mining task is module (or subgraph) identification in biological graphs (e.g., protein–protein or gene co-expression networks) which helps to better understand and treat diseases (183, 184). Finally, graph learning was also used for recommending medication combinations for patients with complex health conditions (185, 186).

Conclusion and future directions

In this paper, we reviewed current state-of-the-art approaches for storing and analyzing datasets with the help of an underlying graph structure. A graph structure is especially useful for representing interconnected data like clinical data. In this context, we highlight several existing applications that exploit the potential of using graphs and graph learning approaches. Furthermore, we classified these applications according to the used graph mining tasks (link prediction, node classification, (sub-)graph classification). This classification is important to find an appropriate graph learning algorithm for the desired use case. It is noteworthy that sometimes one can transfer one or multiple graphs with a specific graph mining task to another graph mining task, e.g., by transferring a single patient node with features (node classification) into a graph with features as nodes (graph classification). The question one needs to ask in advance is which data representation and therefore which graph mining task performs better for a specific use case.

Currently, each application has its individual method (e.g., for data pre-processing, and graph generation) even for the same kind of data (e.g., fMRI data). A generic framework for transferring clinical data into a graph and applying graph learning afterwards, would help to solve several scientific issues quicker and to make the methodology more reproducible. Furthermore, scientists could use previously trained models as a foundation for their use case (transfer learning). Currently, the implementation of such a generic framework is especially difficult due to diverse data formats and the large heterogeneity of clinical data.

Modern graph representation learning techniques such as GNNs consider the input features and thereby decide which features are more important than others. An interesting question might be how much features that are automatically extracted by tools like GNNs differ from features selected by domain experts.

Several advantages of graph databases facilitate storage, retrieval, and exploration of data. First, graph databases provide a higher flexibility in their data schema than relational databases (187). Second, graph query languages like Cypher prevent the formulation of complex Join operations. Third, the easy visualization of complex interconnected graph data (e.g., with Neo4j Bloom (188)) provides a comprehensive overview about the underlying data.

Currently, there are only limited advances in applying machine learning algorithms directly on a graph database, e.g., Neo4j's graph data science library (189) and TigerGraph's data science library (190). Furthermore, to the best of our knowledge, there is no framework that can process graph data from various graph databases with modern graph learning approaches. This strategy is quite promising, because querying only the necessary data might increase the performance for extremely large datasets. Such a graph learning framework could standardize and unify the various query languages of different graph databases (e.g., Cypher and GSQL), integrate state-of-the-art deep learning frameworks like PyTorch (191), TensorFlow (192), Deeplearning4j (193), Microsoft CNTK (194), or flux (195), and allow clear visualizations. A framework using relational databases already exists (196) and showed the potential for easy application of machine learning algorithms and visualizations of the results (197).

However, there are still some other limitations in using graph databases and graph learning algorithms. First, relational databases are much more common in industry making them preferred over graph databases due to their decades-long utilization. Some graph learning algorithms have serious limitations in their performance, and some cannot use features attached to nodes/edges/graphs. Although GNNs seem promising for improving computation efficiency and feature utilization, there are still some problems like over-smoothing, underreaching and over-squashing. Finally, missing standards, diverse formats, lexical disparities, class imbalances, data privacy and the enormous heterogeneity of clinical data makes applying machine learning algorithms even more complex. A combination of implementing further graph learning applications, establishing swarm learning (198) and standardizing data and formats will help to overcome the mentioned problems.

Supplementary material

Supplementary material is available at *Database* online.

Data availability

All required material is contained in the Supplementary material.

Funding

German Research Foundation (DFG) under the project 'Optimizing graph databases focusing on data processing and integration of machine learning for large clinical and biological datasets' [Grant Numbers HE 8077/2-1, SA 465/53-1]).

Conflict of interest

The authors declare that they have no conflict of interests.

Acknowledgements

We thank all co-authors for contributing to the manuscript. Furthermore, we thank the German Research Foundation (DFG) for funding this project [grant numbers HE 8077/2-1, SA 465/53-1].

References

1. Belle,A., Thiagarajan,R., Soroushmehr,S.M.R. *et al.* (2015) Big data analytics in healthcare. *Biomed. Res. Int.*, 370194.
2. Davoudian,A., Chen,L. and Liu,M. (2019) A Survey on NoSQL Stores. *ACM Comput. Surv.*, **51**, 1–43.
3. Park,Y., Shankar,M., Park,B.-H. *et al.* (2014) Graph databases for large-scale healthcare systems: A framework for efficient data management and data services. In: *2014 IEEE 30th International Conference on Data Engineering Workshops*, Chicago, IL, USA. IEEE, pp. 12–19.

4. Angles,R. and Gutierrez,C. (2008) Survey of graph database models. *ACM Comput. Surv.*, **40**, 1–39.
5. Ehrig,H., Prange,U. and Taentzer,G. (2004) Fundamental theory for typed attributed graph transformation. In: Ehrig H (ed) *Graph transformations: Second international conference, ICGT 2004/Hartmut*. 3256. Springer, Berlin, London. pp. 161–177.
6. Färber,M., Bartscherer,F., Menne,C. *et al.* (2017) Linked data quality of DBpedia, freebase, opencyc, wikidata, and YAGO. *SW*, **9**, 77–129.
7. Ehrlinger,L. and Wöß,W. (2016) Towards a definition of knowledge graphs. *Conference: Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16)*, Leipzig, Germany 12 - 15 September 2016.
8. Rotmensch,M., Halpern,Y., Tlimat,A. *et al.* (2017) Learning a health knowledge graph from electronic medical records. *Sci. Rep.*, **7**, 5994.
9. Shi,L., Li,S., Yang,X. *et al.* (2017) Semantic health knowledge graph: semantic integration of heterogeneous medical knowledge and services. *Biomed. Res. Int.*, **2017**, 2858423.
10. Gyrard,A., Gaur,M., Shekarpour,S. *et al.* (2018) Personalized health knowledge graph. In: *CEUR Workshop Proc 2018*, Monterey, USA, October 8th, 2018.
11. Walsh,B., Mohamed,S.K. and Nováček,V. (2020) BioKG. In: d'Aquin M, Dietze S (eds.) *Proceedings of the 29th ACM International Conference on Information & Knowledge Management. Association for Computing Machinery*, New York, NY, pp. 3173–3180.
12. Unni,D.R., Moxon,S.A.T., Bada,M. *et al.* (2022) Biolink Model: A universal schema for knowledge graphs in clinical, biomedical, and translational science. *Clin Transl Sci*, **15**, 1848–1855.
13. Reese,J.T., Unni,D., Callahan,T.J. *et al.* (2021) KG-COVID-19: A framework to produce customized knowledge graphs for COVID-19 response. *Patterns*, **2**, 100155.
14. Domingo-Fernández,D., Baksi,S., Schultz,B. *et al.* (2021) COVID-19 knowledge graph: a computable, multi-modal, cause-and-effect knowledge model of COVID-19 pathophysiology. *Bioinformatics*, **37**, 1332–1334.
15. Wise,C., Ioannidis,V.N., Calvo,M.R. *et al.* (2020) COVID-19 knowledge graph: accelerating information retrieval and discovery for scientific literature. In: *Proceedings of Knowledgeable NLP: the First Workshop on Integrating Structured Knowledge and Neural Networks for NLP, Publisher: Association for Computational Linguistic*, Suzhou, China, December 2020. pp. 1–10.
16. Vrandečić,D. (2012) Wikidata. In: Mille A, Gandon F, Misselis J *et al.* (eds) Proceedings of the 21st international conference companion on World Wide Web. ACM, New York, NY, p. 1063.
17. Mendes,P., Jakob,M. and Bizer,C. (2012) DBpedia: a multilingual cross-domain knowledge base. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), Publisher: European Language Resources Association (ELRA)*, Istanbul, Turkey, pp. 1813–1817.
18. Suchanek,F.M., Kasneci,G. and Weikum,G. (2007) Yago. In: Williamson C, Zurko ME (eds) *WWW 2007: Proceedings of the 16th International Conference on World Wide Web, Banff*, Alberta, Canada, ACM Press, New York. May 8–12, 2007. p. 697.
19. Soussi,R., Aufaure,M.-A. and Baazaoui,H. (2010) Towards social network extraction using a graph database. In: *2010 Second International Conference on Advances in Databases, Knowledge, and Data Applications. Menuires*, France, IEEE. 11-16 April 2010.
20. Warchal L (2012) Using Neo4j Graph Database in Social Network Analysis. *STUDIA INFORMATICA* 33.
21. Miller Justin,J. (2013) Graph database applications and concepts with Neo4j. In: *Southern Association for Information Systems (SAIS) 2013PProceedings*, USA, p. 24.
22. Padgett,J.F. and Ansell,C.K. (1993) Robust action and the rise of the medici, 1400-1434. *Am. J. Sociol.*, **98**, 1259–1319.
23. Yi,N., Li,C., Feng,X. *et al.* (2017) Design and implementation of movie recommender system based on graph database. In: *2017 14th Web Information Systems and Applications Conference (WISA)*. Liuzhou, Guangxi Province, China, IEEE. November 11–12, 2017.
24. Giabelli,A., Malandri,L., Mercorio,F. *et al.* (2021) Skills2Job: A recommender system that encodes job offer embeddings on graph databases. *Appl Soft Comput*, **101**, 107049.
25. Magomedov,S., Pavelyev,S., Ivanova,I. *et al.* (2018) Anomaly detection with machine learning and graph databases in fraud management. *Int. J. Adv. Comput. Sci. Appl.*, **9**.
26. Sadowski,G. and Rathle,P. (2014) Fraud detection: Discovering connections with graph databases. Neo4j. *Neo4j*. US, Germany, Singapore, Sweden, UK. https://we-yun.com/doc/neo4j-.
27. Neo4j Graph Data Platform (2022) *Neo4j graph data platform – the leader in graph databases*. https://neo4j.com/ (27 October 2022, date last accessed).
28. Min,W., Xinglu,Y., Hui,Y. *et al.* (2022) *NebulaGraph: open source and distributed graph database*. https://www.nebula-graph.io/ (27 October 2022, date last accessed).
29. TigerGraph (2022) *Graph analytics platform | graph database | tigerGraph*. https://www.tigergraph.com/ (27 October 2022, date last accessed).
30. *Dgraph | graphql cloud platform (2022) dgraph home*. https://dgraph.io/ (27 October 2022, date last accessed).
31. ArangoDB Oasis (2022) *ArangoDB oasis*. https://www.arangodb.com/ (27 October 2022, date last accessed).
32. Besta,M., Peter,E., Gerstenberger,R. *et al.* (2019) Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries. *Arxiv*. Cornell University. 20th October 2019.
33. Boning,L., Xia,Y., Xie,S. *et al.* (2021) Distance-enhanced graph neural network for link prediction. In: *The 2021 ICML Workshop on Computational Biology (Virtual Conference)*, https://www.semanticscholar.org/paper/Distance-Enhanced-Graph-Neural-Network-for-Link-Li-Xia/732f8f1a44c8994dafa77ed9d2dd0b8661ef7f63.
34. Al_Rabeah Mh, Lakizadeh A (2022) *GNN-DDI: a new data integration framework for predicting drug-drug interaction events based on graph neural networks*. https://www.researchgate.net/publication/361978426_GNN-DDI_A_New_Data_Integration_Framework_for_Predicting_Drug-Drug_Interaction_Events_Based_on_Graph_Neural_Networks (June 2022).
35. Shtar,G., Rokach,L. and Shapira,B. (2019) Detecting drug-drug interactions using artificial neural networks and classic graph similarity measures. *PLoS One*, **14**, e0219796.
36. Jiang,D., Wu,Z., Hsieh,C.-Y. *et al.* (2021) Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *J Cheminform*, **13**, 12.
37. You,J., Liu,B., Ying,Z. *et al.* (2018) Graph convolutional policy network for goal-directed molecular graph generation. In: Bengio S, Wallach H, Larochelle H *et al.* eds. *Advances in Neural Information Processing Systems*. USA. Curran Associates, Inc, Cambridge, MA.
38. Mercado,R., Rastemo,T., Lindelöf,E. *et al.* (2021) Graph networks for molecular design. *Mach. Learn.: Sci. Technol.*, **2**, 25023.
39. Mercado,R., Bjerrum,E.J. and Engkvist,O. (2022) Exploring graph traversal algorithms in graph-based molecular generation. *J Chem Inf Model*, **62**, 2093–2100.
40. Wieder,O., Kohlbacher,S., Kuenemann,M. *et al.* (2020) A compact review of molecular property prediction with graph neural networks. *Drug Discov. Today Technol.*, **37**, 1–12.
41. Hamilton WL *Graph Representation Learning*. 14th edn. Morgan and Claypool, Kentfield, CA, USA.

42. Hamilton,W.L., Ying,R. and Leskovec,J. (2017) Representation learning on graphs: methods and applications. *IEEE Data Eng. Bull*, **40**, 52–74.
43. Lin,X., Quan,Z., Wang,Z.-J. *et al.* (2020) KGNN: knowledge graph neural network for drug-drug interaction prediction. In: desJardins M M, Bessiere C (eds.) *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization*, California, pp. 2739–2745.
44. Liu,Z., Li,X., Peng,H. *et al.* (2020) Heterogeneous similarity graph neural network on electronic health records. In: *2020 IEEE International Conference on Big Data (Big Data). Virtual Event*, IEEE, [S.l.]. 10-13 December 2020, pp. 1196–1205.
45. Li,Y., Qian,B., Zhang,X. *et al.* (2020) Graph neural network-based diagnosis prediction. *Big Data*, **8**, 379–390.
46. Chen,Z., Wang,Y., Zhao,B. *et al.* (2020) Knowledge Graph Completion: A Review. *IEEE Access*, **8**, 192435–192456.
47. Lin,Y., Liu,Z., Sun,M. *et al.* (2015) Learning entity and relation embeddings for knowledge graph completion. In: *AAAI'15: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, pp. 2181–2187.
48. Auten,A., Tomei,M. and Kumar,R. (2020) Hardware acceleration of graph neural networks. In: *2020 57th ACM*, San Francisco, California, USA, IEEE, Piscataway, NJ. pp. 1–6.
49. Heintz,A., Razavimaleki,V., Duarte,J. *et al.* (2020) Accelerated charged particle tracking with graph neural networks on FPGAs. In: *Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020 (Virtual Conference))*, https://arxiv.org/abs/2012.01563
50. Zhang,D., Yin,J., Zhu,X. *et al.* (2017) Network Representation Learning: A Survey. *IEEE Transactions on Big Data*.
51. H-c,Y., You,Z.-H., Huang,D.-S. *et al.* (2022) Graph representation learning in bioinformatics: trends, methods and applications. *Brief. Bioinformatics*, 23.
52. Cai,H., Zheng,V.W. and Chang,K.-C.-C. (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, **30**, 1616–1637.
53. Goyal,P. and Ferrara,E. (2018) Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.*, **151**, 78–94.
54. Ahmedt-Aristizabal,D., Armin,M.A., Denman,S. *et al.* (2022) A survey on graph-based deep learning for computational histopathology. *Comput Med Imaging Graph*, **95**, 102027.
55. Ahmedt-Aristizabal,D., Armin,M.A., Denman,S. *et al.* (2021) Graph-based deep learning for medical diagnosis and analysis: past, present and future. *Sensors (Basel)*, 21.
56. Elshawi,R., Sakr,S., Talia,D. *et al.* (2018) Big data systems meet machine learning challenges: towards big data science as a service. *Big Data Research*, **14**, 1–11.
57. Kumar,A., Boehm,M. and Yang,J. (2017) Data management in machine learning. In: Chirkova R (ed) *Proceedings of the 2017 ACM International Conference on Management of Data. ACM*, New York, NY, pp. 1717–1722.
58. Stonebraker,M. (2010) SQL databases v. NoSQL databases. *Commun. ACM (Association for Computing Machinery)*, **53**, 10–11.
59. Ricardo,C.M. (2003) Structured Query Language. In: *Encyclopedia of Information Systems*. Elsevier, Amsterdam, Netherlands, pp. 279–297.
60. Halpin,T. and Morgan,T. (2008) 12-relational languages Morgan Kaufmann. In: *Information Modeling and Relational Databases*. 2nd edn. Elsevier, Amsterdam, Netherlands, pp. 527–635.
61. Chaudhuri,S., Krishnamurthy,R., Potamianos,S. *et al.* (1995) Optimizing queries with materialized views. In: Yu PSE, Chen ALPE (eds) *Data engineering: 11th International conference: Papers*, Taipei, Taiwan, IEEE Computer Society Press, 6–10 March 1995, pp. 190–200.
62. Roussopoulos,N. (1982) View indexing in relational databases. *ACM Trans. Database Syst.*, **7**, 258–290.
63. Li,Y. and Patel,J.M. (2014) WideTable. *Proc. VLDB Endow.*, **7**, 907–918.
64. Stothers,J.A.M. and Nguyen,A. (2020) Can Neo4j replace postgreSQL in healthcare? *AMIA Jt. Summits Transl. Sci. Proc.*, 646–653.
65. Thomasian,A. (2022) Structured, unstructured, and diverse databases. *Storage Systems*. Elsevier, Amsterdam, Netherlands, pp. 493–563.
66. Angles,R., Arenas,M., Barceló,P. *et al.* (2018) Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, **50**, 1–40.
67. Fernandes,D. and Bernardino,J. (2018) Graph databases comparison: allegrograph, arangoDB, infinitegraph, Neo4J, and OrientDB. In: *Proceedings of the 7th International Conference on Data Science, Technology and Applications*, Porto Portugal SCITEPRESS-Science and Technology Publications, pp. 373–380.
68. Sparsity Technologies *Sparsity-Technologies – Out-of-core graph database for edge computing*. https://www.sparsity-technologies.com/ (27 October 2022, date last accessed).
69. World Wide Web Consortium (2018) *RDF 1.1 concepts and abstract syntax*. https://www.w3.org/TR/rdf11-concepts/ (27 October 2022, date last accessed).
70. ric Miller (Online Computer Library Center), Bob Schloss (IBM),Ora Lassila (Nokia Research Center), Ralph R. Swick (World Wide Web Consortium), Tsuyoshi Sakata (DVL), Murray Maloney In:(Grif), Bob Schloss (IBM), Naohiko URAMOTO (IBM), Bill Roberts (KnowledgeCite) Ron Daniel (LANL), Andrew Layman Chris McConnell (Microsoft), Jean Paoli (Microsoft), R.V. Guha (Netscape), Ora Lassila (Nokia), Ralph LeVan (OCLC), Eric Miller (OCLC), Misha Wolf In:(Reuters), Lauren Wood (SoftQuad), Tim Bray (Textuality), Paul Resnick. (UMich), Tim Berners-Lee (W3C), Dan Connolly (W3C), Jim Miller (W3C), Ralph Swick (W3C) (2017) RDF Model and Syntax.
71. Francis,N., Green,A., Guagliardo,P. *et al.* (2018) Cypher. In: Das G, Jermaine C, Bernstein P (eds) *Proceedings of the 2018 International Conference on Management of Data. ACM*, New York, NY, pp. 1433–1445.
72. Holzschuher,F. and Peinl,R. (2013) Performance of graph query languages. In: Guerrini G (ed) *Proceedings of the Joint EDBT/ICDT 2013 Workshops. ACM*, New York, NY, USA, pp. 195–204.
73. Deutsch,A., Xu,Y., Mingxi,W. *et al.* (2019) TigerGraph: A Native MPP Graph Database. ArXiv abs/1901.08248. In: arxiv.
74. TigerGraph (2022) *GSQL: graph query language | tigerGraph*. https://www.tigergraph.com/gsql/ (16 March 2023, date last accessed).
75. The Apache Software Foundation (2022) *Apache TinkerPop*. https://tinkerpop.apache.org/gremlin.html (27 October 2022, date last accessed).
76. World Wide Web Consortium (2018) *SPARQL 1.1 query language*. https://www.w3.org/TR/sparql11-query/ (27 October 2022, date last accessed).
77. Bronstein,M.M., Bruna,J., LeCun,Y. *et al.* (2017) Geometric deep learning: going beyond euclidean data. *IEEE Signal Process. Mag.*, **34**, 18–42.
78. Kriege,N.M., Johansson,F.D. and Morris,C. (2020) A survey on graph kernels. *Appl. Netw. Sci.*, **5**.
79. Shervashidze,N., Schweitzer,P., van Leeuwen,E.J. *et al.* Weisfeiler-lehman graph kernels. (2011) *J. Mach. Learn. Res.*, **12**, 2539–2561.
80. Vishwanathan,S.V.N., Borgwardt,K.M., Kondor,I.R. *et al.* Graph kernels. *J. Mach. Learn. Res.*, **11**, 1201–1242.
81. Kashima,H., Tsuda,K. and Inokuchi,A. (2003) Marginalized kernels between labeled graphs. In: *ICML'03: Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, pp. 321–328.
82. Borgwardt,K.M. and Kriegel,H. (2005) Shortest-path kernels on graphs. In: *Fifth IEEE International Conference on Data Mining*,

Houston, TX, USA, IEEE/Institute of Electrical and Electronics Engineers Incorporated. pp. 74–81.
83. YANG S. (2013) Networks: an introduction by M. *E. J. Newman. The J. Math Sociol.*, **37**, 250–251.
84. In: Burges,C.J., Bottou,L., Welling,M. *et al.* (eds) (2013) *Advances in Neural Information Processing Systems*. Curran Associates, Inc, Red Hook, NY.
85. Watts,D.J. and Strogatz,S.H. (1998) Collective dynamics of 'small-world' networks. *Nature*, **393**, 440–442.
86. Newman,M.E.J (2018) Networks. Second. Oxford University Press, Oxford.
87. Roweis,S.T. and Saul,L.K. (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**, 2323–2326.
88. Belkin,M. and Niyogi,P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In: *NIPS'01: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, Vancouver British Columbia, Canada, January 2001, pp. 585–591.
89. Ahmed,A., Shervashidze,N., Narayanamurthy,S. *et al.* (2013) Distributed large-scale natural graph factorization. In: Schwabe D (ed) *Proceedings of the 22nd international conference on World Wide Web. Rio de Janeiro, Brazil 13-17 May 2013. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva*, pp. 37–48.
90. Ou,M., Cui,P., Pei,J. *et al.* (2016) Asymmetric transitivity preserving graph embedding. In: Krishnapuram B, Shah M, Smola A *et al*. (eds) *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*, New York, NY, pp. 1105–1114
91. Cao,S., Lu,W. and Xu,Q. (2015) GraRep. In: Bailey J, Moffat A, Aggarwal CC *et al*. (eds) *CIKM'15: Proceedings of the 2015 ACM International Conference on Information and Knowledge Management: October, 19–23, 2015*, Melbourne, Australia., Assocation for Computing Machinery, New York. pp. 891–900.
92. Cai,B., Xiang,Y., Gao,L. *et al.* (2022) *Temporal knowledge graph completion: a survey. In: arxiv*. https://arxiv.org/abs/2201.08236.
93. Brin,S. and Page,L. (1998) The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, **30**, 107–117.
94. Perozzi,B., Al-Rfou,R. and Skiena,S. (2014) DeepWalk. In: Macskassy S, Perlich C, Leskovec J *et al*. (eds.) *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM*, New York, NY, USA, pp. 701–710.
95. Grover,A. and Leskovec,J. (2016) node2vec: scalable feature learning for networks. In: *KDD'16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (San Francisco USA)*, pp. 855–864.
96. Chen,H., Perozzi,B., Hu,Y. *et al.* (2017) HARP: hierarchical representation learning for networks. *Proceedings of the AAAI Conference on Artificial Intelligence*. Palo Alto, California, USA.
97. Cao,S., Lu,W. and Xu,Q. (2016) Deep neural networks for learning graph representations. In: *AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Palo Alto, California, USA, pp. 1145–1152.
98. Wang,D., Cui,P. and Zhu,W. (2016) Structural deep network embedding. In: Krishnapuram B, Shah M, Smola A *et al*. (eds) *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*, New York, NY, pp. 1225–1234.
99. Li,Y., Tarlow,D., Brockschmidt,M. *et al.* (2015) Gated graph sequence neural networks. *Iclr 2016*. Caribe Hilton, San Juan, Puerto Rico.
100. Gallicchio,C. and Micheli,A. (2010) Graph echo state networks. In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*, *Barcelona, Spain, 18–23 July 2010*. IEEE. pp. 1–8.
101. Veličković,P. (2022) *Message passing all the way up. In: arxiv*. https://arxiv.org/abs/2202.11097
102. Veličković,P., Cucurull,G., Casanova,A. *et al.* (2017) Graph attention networks. *Iclr 2018*. Vancouver Convention Center, Vancouver, Canada.
103. Hamilton,W.L., Ying,R. and Leskovec,J. (2017) Inductive representation learning on large graphs. In: *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA.
104. Kipf,T.N. and Welling,M. (2016) Semi-supervised classification with graph convolutional networks. In: *ICLR 2017 (Palais Des Congrès Neptune*. Toulon, France.
105. Xu,K., Hu,W., Leskovec,J. *et al.* (2018) How powerful are graph neural networks? *ICLR 2019 (Ernest N.* Morial Convention Center, New Orleans.
106. Zhang,C., Song,D., Huang,C. *et al.* (2019) Heterogeneous graph neural network. In: Teredesai A, Kumar V, Li Y *et al.* (eds.) *KDD '19. Association for Computing Machinery*, New York, pp 793–803.
107. Wang,X., Ji,H., Shi,C. *et al.* Heterogeneous graph attention network. In: *WWW '19: The World Wide Web Conference (Association for Computing Machinery*, New York, United States), May 2019, pp. 2022–2032.
108. Chen,D., Lin,Y., Li,W. *et al.* (2019) Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Aaai 2020*. New York, New York, USA.
109. Alon,U. and Yahav,E. (2020) On the bottleneck of graph neural networks and its practical implications*Iclr 2021*. Vienna, Austria.
110. Tang,J., Qu,M., Wang,M. *et al.* (2015) LINE: Large-scale Information Network Embedding. In: *International World Wide Web Conference Committee (IW3C2) WWW 2015*, May 18–22, 2015, Florence, Italy.
111. Wang,H., Wang,J., Wang,J. *et al.* (2017) GraphGAN: graph representation learning with generative adversarial nets. In: *AAAI'18/IAAI'18/EAAI'18: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence; Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, New Orleans, Louisiana, USA, 306. pp. 2508–2515.
112. Zhang,M., Li,P., Xia,Y. *et al.* (2020) Labeling trick: a theory of using graph neural networks for multi-node representation learning. In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, Virtual Conference.
113. Duvenaud,D., Maclaurin,D., Aguilera-Iparraguirre,J. *et al.* (2015) Convolutional networks on graphs for learning molecular fingerprints. In: *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing*, Palais des Congrès de MontréalMontréal CANADA, pp. 2224–2232.
114. Defferrard,M., Bresson,X. and Vandergheynst,P. (2016) convolutional neural networks on graphs with fast localized spectral filtering. In: *30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain.
115. Bruna,J., Zaremba,W., Szlam,A. *et al.* (2013) Spectral networks and locally connected networks on graphs.
116. Li,J., Cai,D. and He,X. (2017) *Learning Graph-Level Representation for Drug Discovery*. arxiv.
117. van der Maaten,L. and Hinton,G. (2008) Visualizing data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.
118. Maćkiewicz,A. and Ratajczak,W. (1993) Principal components analysis (PCA). *Comput Geosci*, **19**, 303–342.
119. Lü, L. and Zhou,T. (2011) Link prediction in complex networks: A survey. *Phys. A: Stat. Mech. Appl.*, **390**, 1150–1170.
120. Almansoori,W., Gao,S., Jarada,T.N. *et al.* (2012) Link prediction and classification in social networks and its application in healthcare and systems biology. *Netw. Model Anal. Health Inform. Bioinforma.*, **1**, 27–36.
121. Zhou,J., Cui,G., Hu,S. *et al.* (2020) Graph neural networks: A review of methods and applications. *AI Open*, **1**, 57–81.
122. von,L.U. (2007) A tutorial on spectral clustering. In: *Statistics and Computing*. Springer Link, pp. 395–416.

123. Zitnik,M., Agrawal,M. and Leskovec,J. (2018) Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, **34**, i457–i466.
124. Lu,Y., Guo,Y. and Korhonen,A. (2017) Link prediction in drug-target interactions network using similarity indices. *BMC Bioinform.*, **18**, 39.
125. Lei,C. and Ruan,J. (2013) A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity. *Bioinformatics*, **29**, 355–364.
126. Folino,F. and Pizzuti,C. (2012) Link prediction approaches for disease networks. In: Böhm C (ed) *Information technology in bio- and medical informatics: Third International Conference, ITBAM 2012*, Vienna, Austria, pp. 99–108.
127. Gao,M., Lu,J. and Chen,F. (2022) Medical knowledge graph completion based on word embeddings. *Information*, **13**, 205.
128. Ma,T., Xiao,C., Zhou,J. *et al.* (2018) Drug similarity integration through attentive multi-view graph auto-encoders. In: *Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, Stockholmsmässan´, Stockholm.
129. Bordes,A., Usunier,N., Garcia-Durán,A. *et al.* (2013) Translating embeddings for modeling multi-relational data. In: *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. Lake Tahoe, Nevada, United States.
130. Wang,Z., Zhang,J., Feng,J. *et al.* (2014) Knowledge Graph Embedding by Translating on Hyperplanes. In: *AAAI'14: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Québec City, Québec, Canada, pp. 1112–1119.
131. Yang,B., Yih,W., He,X. *et al.* (2014) Embedding entities and relations for learning and inference in knowledge bases. *Iclr 2015*. San Diego, CA, USA.
132. Socher,R., Chen,D., Manning,C.D. *et al.* (2013) Reasoning With Neural Tensor Networks for Knowledge Base Completion. In: Burges CJ, Bottou L, Welling M *et al* eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc, Lake Tahoe Nevada, California, USA.
133. Trouillon,T., Welbl,J., Riedel,S. *et al.* (2016) Complex Embeddings for Simple Link Prediction.
134. Yang,S., Tian,J., Zhang,H. *et al.* (2019) TransMS: knowledge graph embedding for complex relations by multidirectional semantics. In: Kraus S (ed) *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19). International Joint Conferences on Artificial Intelligence*, California, pp. 1935–1942.
135. Yue,X., Wang,Z., Huang,J. *et al.* (2020) Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics*, **36**, 1241–1251.
136. Parisot,S., Ktena,S.I., Ferrante,E. *et al.* (2018) Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease. *Med Image Anal*, **48**, 117–130.
137. Huang,Y. and Chung,A.C.S. (2020) Edge-variational Graph Convolutional Networks for Uncertainty-aware Disease Prediction. In: *MICCAI 2020: Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*. Lima, Peru, pp. 562–572.
138. Vivar,G., Kazi,A., Burwinkel,H. *et al.* (2020) Simultaneous imputation and classification using Multigraph Geometric Matrix Completion (MGMC): Application to neurodegenerative disease classification. **117**.
139. Kazi,A., Shekarforoush,S., Arvind Krishna,S. *et al.* (2019) Graph Convolution Based Attention Model for Personalized Disease Prediction. In: Shen D, Liu T, Peters TM, Zhou S, Yap P-T, Khan A (eds.) Springer, Cham, Switzerland, pp. 122–130.
140. Wang,X., Yao,L., Rekik,I. *et al.* (2022) Contrastive Graph Learning for Population-based fMRI Classification. *MICCAI 2022: Medical Image Computing and Computer Assisted Intervention – MICCAI 2022 (Resort World Convention Centre Singapore)*, 221–230.
141. Wang,D., Lei,C., Zhang,X. *et al.* (2021 - 2021) Identification of Depression with a Semi-supervised GCN based on EEG Data. In: *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, pp. 2338–2345.
142. Anklin,V., Pati,P., Jaume,G. *et al.* (2021) Learning Whole-Slide Segmentation from Inexact and Incomplete Labels using Tissue Graphs. *Med. Image Comput. Comput. Assist. Interv*, 636–646.
143. Ma,T. and Zhang,A. (2018) AffinityNet: semi-supervised few-shot learning for disease type prediction.
144. Fan,J., Cannistra,A., Fried,I. *et al.* (2017) A Multi-Species Functional Embedding Integrating Sequence and Network Structure. *biorxiv*.
145. Lian,Q., Qi,Y., Pan,G. *et al.* (2020) Learning graph in graph convolutional neural networks for robust seizure prediction. *J. Neural. Eng.*, **17**, 35004.
146. Covert,I., Krishnan,B., Najm,I. *et al.* (2019) Temporal Graph Convolutional Networks for Automatic Seizure Detection. In: *Proceedings of Machine Learning Research 106 (JMLR Workshop and Conference Proceedings)*. pp. 1–19.
147. Mathur,P. and Chakka,V.K. (2020) Graph Signal Processing of EEG signals for Detection of Epilepsy. In: *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. Noida, India. 27-28 February 2020. IEEE, [S.l.], pp. 839–843.
148. Wang,J., Gao,R., Zheng,H. *et al.* (2022) SSGCNet: A Sparse Spectra Graph Convolutional Network for Epileptic EEG Signal Classification. In: *IEEE Transactions on Neural Networks and Learning Systems*. pp. 1–15.
149. Jang,S., Moon,S.-E. and Lee,J.-S. (2018) EEG-based video identification using graph signal modeling and graph convolutional neural network. In: *Conference: ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Rodos Palace, Greece.
150. Jang,S., Moon,S.-E. and Lee,J.-S. (2019) EEG-based Emotional Video Classification via Learning Connectivity Structure. *IEEE Trans. Affect.*,**14**, 1586–1597.
151. Wang,Z.-M., Zhou,R., He,Y. *et al.* (2020) Functional Integration and Separation of Brain Network Based on Phase Locking Value During Emotion Processing. *IEEE Trans Cogn Dev Syst (United States)*, p. 1.
152. Wang,X., Zhang,T., Xu,X. *et al.* (2018) EEG Emotion Recognition Using Dynamical Graph Convolutional Neural Networks and Broad Learning System. In: Zheng H (ed) *2018 IEEE International Conference on Bioinformatics and Biomedicine*. Madrid, Spain: IEEE, [Piscataway, NJ], pp. 1240–1244.
153. Song,T., Zheng,W., Song,P. *et al.* (2020) EEG Emotion Recognition Using Dynamical Graph Convolutional Neural Networks. *IEEE Trans. Affect. Comput*, **11**, 532–541.
154. Wang,Z., Tong,Y. and Heng,X. (2019) Phase-Locking Value Based Graph Convolutional Neural Networks for Emotion Recognition. *IEEE Access*, **7**, 93711–93722.
155. Yin,Y., Zheng,X., Hu,B. *et al.* (2021) EEG emotion recognition using fusion model of graph convolutional neural networks and LSTM. *Appl Soft Comput*, **100**, 106954.
156. Zhong,P., Wang,D. and Miao,C. (2019) EEG-Based Emotion Recognition Using Regularized Graph Neural Networks. In: *IEEE Transactions on Affective Computing*. IEEE.
157. Liu,S., Zheng,W., Song,T. *et al.* Sparse Graphic Attention LSTM for EEG Emotion Recognition. In: Gedeon T, Wong KW, Lee M, Tom G, Kok Wai W, Minho L (eds.) vol. 1142. Springer, Cham, Switzerland, pp. 690–697.
158. Jia,Z., Lin,Y., Wang,J. *et al.* (2020) GraphSleepNet: Adaptive Spatial-Temporal Graph Convolutional Networks for Sleep Stage Classification. In: desJardins M, Bessiere C (eds.) *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization*. California, pp. 1324–1330.
159. Supekar,K., Menon,V., Rubin,D. *et al.* (2008) Network analysis of intrinsic functional brain connectivity in Alzheimer's disease. *PLoS Comput. Biol.*, **4**, e1000100.

160. Gopinath,K., Desrosiers,C. and Lombaert,H. (2019) Learnable Pooling in Graph Convolution Networks for Brain Surface Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, **44**, Issue 864–876.
161. Yang,H., Li,X., Wu,Y. *et al.* (2019) Interpretable Multimodality Embedding of Cerebral Cortex Using Attention Graph Network for Identifying Bipolar Disorder. In: Shen D, Liu T, Peters TM *et al.* (eds.) *Medical image computing and computer assisted intervention—MICCAI 2019: 22nd International Conference*, Shenzhen, China, Part III/Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, Ali Khan (eds.). Springer, Cham, Switzerland, pp. 799–807.
162. Li,X., Dvornek,N.C., Zhou,Y. *et al.* Graph Neural Network for Interpreting Task-fMRI Biomarkers.
163. Xing,X., Li,Q., Wei,H. *et al.* (2019) Dynamic Spectral Graph Convolution Networks with Assistant Task Training for Early MCI Diagnosis. In: Shen D, Liu T, Peters TM, Zhou S, Yap P-T, Khan A (eds.) vol. 11767. Springer, Cham, Switzerland, pp. 639–646.
164. McDaniel,C. and Quinn,S. (2019) Developing a Graph Convolution-Based Analysis Pipeline for Multi-Modal Neuroimage Data: An Application to Parkinson's Disease. In: *Proceedings of the 18th Python in Science Conference*. SciPy Austin, Texas 8. pp. 42–49.
165. Li,X., Zhou,Y., Dvornek,N. *et al.* (2021) BrainGNN: interpretable brain graph neural network for fMRI analysis. *Med Image Anal*, **74**, 102233.
166. Kim,B.-H., Ye,J.C. and Kim,-J.-J. (2021) Learning dynamic graph representation of brain connectome with spatio-temporal attention. *NeurIPS 2021*, Virtual Event.
167. Zhang,X., Tokoglu,F., Negishi,M. *et al.* (2011) Social network theory applied to resting-state fMRI connectivity data in the identification of epilepsy networks with iterative feature selection. *J. Neurosci. Methods*, **199**, 129–139.
168. Adnan,M., Kalra,S. and Tizhoosh,H.R. (2020) Representation Learning of Histopathology Images using Graph Neural Networks. In: *Conference: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA.
169. Zheng,Y., Jiang,B., Shi,J. *et al.* (2019) Encoding Histopathological WSIs Using GNN for Scalable Diagnostically Relevant Regions Retrieval. In: Shen D, Liu T, Peters TM, Dinggang S, Tianming L, Terry M, Peters, Lawrence H, Staib, Zhou S, Yap P-T, Khan A (eds.) Vol. 11764. Springer, Cham, Switzerland, pp. 550–558
170. Rhee,S., Seo,S. and Kim,S. (op. 2018) Hybrid Approach of Relation Network and Localized Graph Convolutional Filtering for Breast Cancer Subtype Classification. In: Lange J (ed) *IJCAI. International Joint Conferences on Artificial Intelligence, [S. l.]*, pp 3527–3534.
171. Pati,P., Jaume,G., Foncubierta,A. *et al.* (2021) Hierarchical Graph Representations in Digital Pathology. In: *Medical Image Analysis*. Science Direct (Elsevier), Amsterdam, Netherlands, p. 102264.
172. Wu,J., Zhong,J.-X., Chen,E.Z. *et al.* (2019) Weakly- and Semi-supervised Graph CNN for Identifying Basal Cell Carcinoma on Pathological Images. In: Zhang D, Zhou L, Jie B *et al.* (eds) *Graph Learning in Medical Imaging: First International Workshop, GLMI 2019, held in conjunction with MICCAI 2019*. Shenzhen, China, 2019, Proceedings/Daoqiang Zhang, Luping Zhou, Biao Jie, Mingxia Liu (eds.) 11849. Springer Cham, Switzerland, pp. 112–119.
173. Wang,J., Chen,R.J., Lu,M.Y. *et al.* (2019) Weakly Supervised Prostate TMA Classification via Graph Convolutional Networks. In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. Iowa City, Iowa, USA
174. Studer,L., Wallau,J., Dawson,H. *et al.* (2021) Classification of Intestinal Gland Cell-Graphs Using Graph Neural Networks. In: Vezzani R (ed.) *Proceedings of ICPR 2020: 25th International Conference on Pattern Recognition: Milan*. IEEE, Piscataway, NJ, pp. 3636–3643.
175. Zhao,Y., Yang,F., Fang,Y. *et al.* Predicting Lymph Node Metastasis Using Histopathological Images Based on Multiple Instance Learning With Deep Graph Convolution. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13–19 June 2020 Seattle, WA, USA, pp. 4836–4845.
176. Yu,X., Lu,S., Guo,L. *et al.* (2021) ResGNet-C: A graph convolutional neural network for detection of COVID-19. *Neurocomputing*, **452**, 592–605.
177. S-y,L., Zhang,Z., Zhang,Y.-D. *et al.* (2021) CGENet: A Deep Graph Model for COVID-19 Detection Based on Chest CT. *Biology (Basel)*, 11.
178. Fout,A., Byrd,J., Shariat,B. *et al.* (2017) Protein interface prediction using graph convolutional networks. In: *31st Conference on Neural Information Processing Systems (NIPS 2017)*. Long Beach, CA, USA.
179. Stankevičiūtė,K., Azevedo,T., Campbell,A. *et al.* (2020) *Population Graph GNNs for Brain Age Prediction*. biorxiv.
180. Choi,E., Xu,Z., Li,Y. *et al.* (2019) Learning the Graphical Structure of Electronic Health Records with Graph Convolutional Transformer. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*. New York, USA.
181. Chen,R.J., Lu,M.Y., Wang,J. *et al.* (2022) Pathomic Fusion: An Integrated Framework for Fusing Histopathology and Genomic Features for Cancer Diagnosis and Prognosis. *IEEE Trans. Med. Imaging*, **41**, 757–770.
182. Levy,J., Haudenschild,C., Barwick,C. *et al.* (2021) Topological Feature Extraction and Visualization of Whole Slide Images using Graph Neural Networks. In: Altman R (ed.) *Biocomputing 2021: Proceedings of the Pacific Symposium*. World Scientific Publishing, Singapore, pp. 285–296.
183. Sureka,M., Patil,A., Anand,D. *et al.* (2020) Visualization for Histopathology Images using Graph Convolutional Neural Networks. In: *IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*. Cincinnati, OH, USA.
184. Choobdar,S., Ahsen,M.E., Crawford,J. *et al.* (2019) Assessment of network module identification across complex diseases. *Nat. Methods*, **16**, 843–852.
185. Cowen,L., Ideker,T., Raphael,B.J. *et al.* (2017) Network propagation: a universal amplifier of genetic associations. *Nat. Rev. Genet.*, **18**, 551–562.
186. Shang,J., Ma,T., Xiao,C. *et al.* (2019) Pre-training of Graph Augmented Transformers for Medication Recommendation. *IJCAI2019*. Macao, China.
187. Shang,J., Xiao,C., Ma,T. *et al.* (2018) GAMENet: Graph Augmented MEmory Networks for Recommending Medication Combination. *Aaai 2019*. Honolulu, Hawaii, USA.
188. Batra,S. (2013) Comparative analysis of Relational and Graph databases. In: *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*. Riyadh, Saudi Arabia). https://ieeexplore.ieee.org/document/8441982.
189. Neo4j Graph Data Platform (2022) *Bloom*. https://neo4j.com/product/bloom/ (18 Nov 2022, date last accessed).
190. Neo4j Graph Data Platform (2022) *The Neo4j Graph Data Science Library Manual v2.2 - Neo4j Graph Data Science*. https://neo4j.com/docs/graph-data-science/current/ (28 Oct 2022, date last accessed).
191. TigerGraph (2022) *TigerGraph Graph Algorithms | Obtain Insights at Scale*. https://www.tigergraph.com/graph-data-science-library/ (28 October 2022, date last accessed).
192. Paszke,A., Gross,S., Massa,F. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Vancouver Convention Center, Vancouver, CANADA. December 2019. Article No: 721, pp. 8026–8037.
193. Abadi,M., Barham,P., Chen,J. *et al.* (2016) TensorFlow: A System for Large-Scale Machine Learning, Savannah, GA,

USA. In: *OSDI'16: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, pp. 265–283
194. Deeplearning4j (2022) *Deeplearning4j Suite Overview*. https://deeplearning4j.konduit.ai/ 28 October 2022, date last accessed
195. Seide,F. and Agarwal,A. (2016) CNTK. In: Krishnapuram B, Shah M, Smola A *et al.* (eds) *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. New York, NY, p. 2135
196. Innes,M., Saba,E., Fischer,K. *et al.* (2018) Fashionable Modelling with Flux. In: *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, *Montréal, Canada*.
197. MindsDB (2022) *Machine Learning In Your Database Using SQL*. https://mindsdb.com/ (28 October 2022, date last accessed).
198. Nascimento,M. and Lopes,P. (2022) Internet of things and machine learning applied to the thermal comfort of internal environments. *Indoor Built Environ.*, **31**, 2274–2290.
199. Warnat-Herresthal,S., Schultze,H., Shastry,K.L. *et al.* (2021) Swarm Learning for decentralized and confidential clinical machine learning. *Nature*, **594**, 265–270.
200. Vonbach,P., Dubied,A., Krähenbühl,S. *et al.* (2008) Prevalence of drug-drug interactions at hospital entry and during hospital stay of patients in internal medicine. *Eur. J. Intern. Med.*, **19**, 413–420.
201. McFarlane,S.I., Kumar,A. and Sowers,J.R. (2003) Mechanisms by which angiotensin-converting enzyme inhibitors prevent diabetes and cardiovascular disease. *Am. J. Cardiol.*, **91**, 30H–37H.
202. Kanehisa,M. and Goto,S. (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **28**, 27–30.
203. McCallum,A.K., Nigam,K., Rennie,J. *et al.* (2000) Automating the Construction of Internet Portals with Machine Learning. *Inf Retr Boston*, **3**, 127–163.
204. Fey,M. and Lenssen,J.E. (2019) Fast Graph Representation Learning with PyTorch Geometric. In: *ICLR 2019 (RLGM Workshop)*, *New Orleans, Louisiana, USA*. https://arxiv.org/abs/1903.02428.