Learning with Graph Neural Networks

Koffka Khan
Department of Computing and Information Technology, Faculty of Science and Agriculture, The University of the West Indies, St. Augustine Campus, TRINIDAD AND TOBAGO
Email address: koffka.khan@gmail.com

Abstract— *In this paper I will discuss machine learning with graphs and its different use cases. Machine-learning prediction challenges at the node level, edge level, and graph level were discussed. The selection of a graph representation was then discussed in terms of directed and undirected graphs, bipartite graphs, weighted and unweighted graphs, adjacency matrices, and some definitions from graph theory, such as the connectivity of graphs, weak connectivity, and strong connectivity, as well as the concept of node degree*.

Keywords— *Machine learning: node: edge: graph: directed: undirected: bipartite: weighted: unweighted: adjecency matrix.*

I. Introduction

The goal of the first section is to motivate and excite you with graphs, structured data and the unique machine learning techniques we can use on it. So why use graphs? A generic language for describing and analyzing entities with relationships and interactions is represented by graphs. This means that we should truly think of the world or a particular area in terms of networks and relations between these things rather than as a collection of isolated data points. This indicates that there is an underlying graph of relationships between the entities, and that these entities are related to one another in accordance with these connections or the graph's structure.

There are many different types of data that can naturally be represented as graphs and modeling these graphical relationships and relational structure of the underlying domain enables us to develop models of the underlying phenomena that underlie the data that are considerably more truthful and accurate [10]. Therefore, for instance, we can consider that infrastructure, events, disease pathways, networks of particles in physics, networks of creatures in food webs, and networks of computer networks can all be represented as graphs. In a similar way, we might consider how our brain's neurons are connected, as well as social networks, economic networks, communication networks, patients between different papers, and the Internet as a massive communication network.

Once more, all of these fields are by nature networks or graphs. In addition, that representation enables us to capture the connections among various things or entities in these many domains. Finally, facts can be represented as relationships between various entities using knowledge. The regulatory systems in our cells can be characterized as operations controlled by connections between various elements.

Even real-life scenes from the outside world can be represented as graphs showing the relationships between the scene's objects [29]. They are referred to as scene graphs. Software can be represented as a graph of calls between various functions [14], for example, or as the structure of the code as captured by an abstract syntax tree.

As a set of graphs, we can naturally take molecules, which are made up of nodes, atoms, and bonds, and represent the atoms as nodes and the relationships between them as edges [22]. Of course, we can take three-dimensional shapes and represent them as graphs in computer graphics [5]. Therefore, in all of these fields, graphical structure is crucial because it enables us to accurately characterize the underlying domain and events.

There are two large sections of data that can be represented as graphs, which is how we will conceptualize graph relational data. First, there are what are known as natural graphs or networks, which naturally describe underlying domains as graphs. For instance, graphs naturally arise in social networks [34], societies, which are collections of seven billion people connected through connections, conversations, and transactions between technological devices, such as phone calls and financial transactions. In biomedicine, genes and proteins that control biological processes exist. We can use a graph to visualize connections between these many biological entities [15]. As previously stated, our brains' connections between neurons are essentially a network of links. Present these domains as networks if you want to model them.

Another example of a domain with relational structure is one where we can express the relational structure using graphs. Information and knowledge, for instance, are frequently arranged and connected. A graph can be used to represent software. We frequently have the ability to connect data elements that are comparable. In addition, this will produce our similarity network, or graph. Moreover, in physics, we can use particle-based simulation to model how particles are related to one another through and they depict this with the graph. Other domains with natural relational structure include molecules, scene graphs [13], and 3D shapes.

This indicates that there are numerous distinct domains, as well as other domains that can naturally be depicted as graphs to capture the relationship structure, such as, natural networks or graphs. We will discuss how to make use of this relational structure to be able to produce better, more accurate predictions as our main topic. In addition, it is particularly crucial now because couplings domains have developed a relational structure that can be represented by a graph. Furthermore, by formally modeling these interactions, we will be able to improve performance, create more accurate models, and make predictions that are more precise.

In addition, in the era of deep learning [23], where the modern toolkit for deep learning is focused on simple data types, this is particularly intriguing and significant. It is focused on simple grids and sequences. A series, like text or speech, has a linear structure, and wonderful tools have been created to evaluate this kind of structure. All images have this spatial locality and the ability to be resized, making it possible to express them as fixed size grids or standards. In addition, once more, deep learning techniques have proven to be quite effective at processing these kinds of fixed-size photos.

However, because they are more complicated, networks, graphs, etc., are far more difficult to process. They are firstly arbitrary in size and complex in topology. Additionally, there is no spatial localization like in grids or text. We are familiar with left and right in text, up and down in grids, and left and right. However, there is no reference point or concept of spatial locality in networks. The second crucial aspect is that deep learning cannot be performed since there is no set node ordering or reference point. Additionally, these networks frequently contain multi-model properties and are dynamic.

We will therefore focus on how to create neural networks that are much more broadly applicable [2]. How can we create neural networks that can process intricate data structures like graphs? Moreover, in reality, the newest area of study in deep learning and representational learning is relational data graphs. Accordingly, what we would like to do is develop neural networks, but instead of using our graph as input, we will use it as output so that they may make predictions.

Moreover, these predictions can be made at the level of individual nodes, pairs of nodes, or linkages, or they can be much more sophisticated, such as a newly constructed graph or a prediction of a particular molecule's attribute that can be represented as a graph on the input.

In addition, the question is, how do we create the neural network architecture that will enable us to complete this process from beginning to end without the need for human feature engineering?

What I am trying to say is that traditional machine learning approaches put a lot of work into defining appropriate features and ways to capture the data's structure so that machine-learning models can utilize it. Therefore, our main focus in this section will be on representation learning, which skips the feature engineering stage. In essence, we can automatically learn a good representation of the graph so that it can be utilized for downstream machine learning algorithms after we get our graph data.

Automatically extracting or learning features from the graph is what this learning is all about. We can conceive of representation learning as mapping graph nodes to d-dimensional vectors or embeddings [4] so that nodes that appear to be part of the network are embedded closely together in the embedding space. The objective is to learn a function f that will take the nodes and convert them into these d-dimensional, real valued vectors. This vector will be called a representation, a feature representation, an embedding of a particular node, an embedding of a given graph, an embedding of a certain link, etc.

This paper consists of four sections. In Section II we give some applications in graph machine learning. In Section III we discuss the selection of graph representation, while the conclusion in given in Section VI.

II. Applications

I am going to talk about graph machine learning applications in this section of the paper and how it affects a variety of applications. We can create many task kinds in machine learning on graphs. Tasks can be created down to the level of specific nodes, or at the level of edges, which are pairings of nodes, we can define jobs. Both jobs at the level of entire graphs, such as graph level prediction or graph generation, as well as tasks at the level of subgraphs of nodes can be identified or defined.

Next, I am going to walk you through these many task levels and demonstrate the various domains and applications that this kind of modeling methodology may be used in. Therefore, when discussing node level jobs, we typically refer to node classification, where we are attempting to predict a node's attribute. For instance, categorize products or internet folks. In link prediction, we attempted to foretell the presence or absence of missing links between two nodes. The completion of knowledge graphs is one such example of this endeavor. In a task at the graph level called "graph classification," we endeavor to classify various graphs.

For instance, we might want to visualize molecules as graphs and then forecast their attributes. This is a particularly fascinating and crucial task for drug creation since it requires us to predict the characteristics of various compounds and medications. We may also perform clustering or community detection, where the objective is to find neatly separated areas of the graph where nodes are highly or densely related to one another. In addition, one use for these would be social circle detection. There are other kinds of tasks, too. For instance, graph evolution or graph generation, where graph generation might be used, for instance, to identify new chemical structures for drugs [9].

Moreover, in physics, where we want to perform precise simulations of different sorts of physics phenomena, which can be represented as a graph. In this field, forecasting graphs and its evolution is quite beneficial. Therefore, in all of these machine-learning activities, we use graphs, which results in high-impact applications. I want to offer you a few examples of them.

I am going to start by providing you with some examples of node level machine learning applications. The following issue is a recent one. Protein folding [28] is the process by which molecules known as proteins regulate various biological processes in our bodies. For instance, drugs work by binding to or altering the behavior of various proteins, which then alters the biological processes in our bodies and for example, causes us to be cured or healed. Amino acids make up proteins. Our protein can also be thought of as a series of amino acids. However, because of magnetic and other types of forces, these proteins are not actually chains or strains; instead, they fold into a variety of intricate forms.

Can you anticipate the 3D structure of the underlying protein from a sequence of amino acids? This is one of the most crucial biological puzzles that has not been solved. Therefore, the computational challenge that scientists have been holding competitions over since the 1970s, is how to compute or computationally forecast a protein's three-dimensional structure based only on its amino acid sequence. The three-dimensional structures of an antibody protein [33] are shown below, see Figure 1. As you can see, the folding of a protein is quite complex and depends on the arrangement of its amino acids.

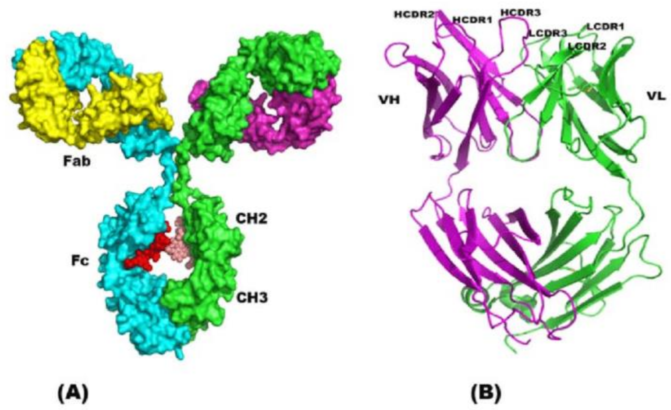

Fig. 1. (A) Three-dimensional structure of antibody structure (protein data bank code: Igg1.ent). Antibody is a Y-shaped molecule with two arms (Fabs) and a stem (Fc region). These two domains are connected by disulfide links. The linkers allow a flexible movement in the antibody. Carbohydrates in the Fc region are shown as small red and pink spheres. (B) Antigen binding domain, Fab is shown in ribbon representation. Light and heavy chains are shown in green and purple, respectively. Fab domain is characterized by β-strands sandwiched as shown and interleaved with loops called complementary determining region (CDR). Six CDR loops mediate antigen specificity and binding. [26]

So the question is, can we predict the three-dimensional structure of the protein given a sequence of amino acids? In addition, the solution to this issue was just recently found. DeepMind released AlphaFold [8] in the middle of December of 2020, which enhanced the performance or accuracy of this application for protein folding by 30% and went as high as values in the upper 90s.

The crucial concept that enabled this breakthrough in science, artificial intelligence, and machine learning was to visualize the underlying protein as a graph. They depicted it as a spatial graph, with the amino acids in the protein sequence acting as the nodes, and amino acids that are spatially close to one another acting as the edges. This indicates that the graph neural network technique was trained to predict the new positions of the- of the amino acids given the positions of all the amino acids and the edges proximities between them. By simulating protein folding in this manner, it was possible to anticipate the final positions of the molecules as well as their positions at the time of their final folding.

Therefore, the utilization of graph representation and the graph neural network technology was a major component in creating this work, in making this scientific achievement in protein folding. This, however, was at the node level, where we essentially attempted to anticipate the position of each node in the graph in order to ascertain its three-dimensional arrangement in a protein.

We will now discuss edge-level machine learning tasks [18], where we essentially perform link prediction or attempt to comprehend the relationships between various nodes. The first instance of this is in recommender systems, which are essentially thought of as people engaging with goods, such as products, movies, songs, and other media. There will be two different kinds of nodes. Both users and products would be present. If a person consumes, purchases, reviews, or watches a certain movie or song, there is an edge between them and the object. We would like to forecast or suggest additional items that specific users may be interested in in the future based on the structure of this graph, the characteristics of the users, and the objects. Thus, a bipartite graph and a graph issue are both natural outcomes.

Modern recommender systems are built on these graphical representations and use graph representation learning and graph neural networks to create predictions. These systems are used in organizations like Pinterest, LinkedIn, Facebook, Instagram, Alibaba, and elsewhere [32]. The important realization is that we can basically learn how to represent or embed nodes of this graph so that connected nodes are embedded nearer to one another than unconnected nodes. For example, in the case of Pinterest, we might see the images as nodes in a graph, with the objective being to embed similar nodes—i.e., related images—closer together than unrelated ones.

One way to accomplish this is to build a network that is bipartite [6], with photos on top and, for instance, people or Pinterest boards at the bottom. Then we can construct a neural network approach that will take the attribute information of these various pins—basically the content of the image—and change it across the underlying graph to produce a robust embedding of a specific image. It turns out that this strategy performs enormously better than thinking about images alone.

As a result, graph structure combined with photos produces considerably better suggestions than just the images alone. Therefore, in this task example, understanding relationships between pairs of nodes or pairs of images is achieved by essentially stating that related nodes should be embedded closer together and that their distance from one another should be smaller than that between pairs of images that are unrelated to one another. A completely different example of a link level prediction task follows.

This is about adverse effects of medication combinations. The issue is that many people take many medications concurrently to manage complicated and comorbid conditions.

As an illustration, over fifty percent of Americans over the age of 70 currently use four, five, or more medicines concurrently [16]. In addition, many individuals take 20 or more medications to treat a variety of complex comorbid conditions. For instance, a person who experiences insomnia, depression, and heart disease all at the same time will take a variety of medications all at once. The issue is that these medications interact with one another, which produces new unwanted side effects. In essence, drug interactions cause more illnesses or problems for that person, such as additional diseases or disorders.

Additionally, since there are too many distinct drug combinations, we are unable to examine each one experimentally or in clinical trials to see what adverse effects they might cause. Therefore, the challenge is: Can we create a predictive engine that, for any given pair of medications, can foretell their potential interactions and potential side effects? Additionally, this is a graph problem. So allow me to explain the formulation process.

We build a two-level heterogeneous network [31] where triangles represent the various medications, and the circles represent the proteins in our bodies. Then, the many proteins are the targets of how medications function. Therefore, the edges are where triangles and circles meet. The protein-protein interaction network has been mapped out by biologists, who use experiments to determine whether two proteins physically interact to control a particular biological process or function. We also understand which proteins interact with one another through experimental research. Furthermore, this is referred to as a protein-protein interaction network.

The final set of linkages in this graph pertain to known side effects. For instance, the link between nodes (e.g. D and E) indicates that the side effect of type (S) is known to occur when these two medications are taken together. Of course, this network of side-effects may be infamously lacking in connections. So the challenge is: can we infer or predict the missing edges and connections in this network that, in essence, would tell us what kinds of side effects we could anticipate if we take, or if a person takes two medications at once?

Therefore, the way we think of this is as a connection prediction between triangle nodes of the graph, where the core question is, given the two medications, what kind of side effects, if any, might be expected. What's more intriguing is that you may use this technique very precisely to find novel adverse effects that were not previously known.

For instance, in this case the model output the top ten predictions it is most confident in, which essentially read as, "If you consider these two pills, then this specific side effect is likely to occur."

Additionally, none of these side effects is listed in the FDA's official database. To find reports that could tell us whether and provide evidence as to whether this particular pair of medications could result in a given side effect, the authors picked the top 10 predictions from the model and searched the medical literature and clinical medical notes. Then, for the top five rankings out of the top 10, we discovered that there is some scientific evidence suggesting that these forecasts may actually be accurate.

Therefore, these were the machine learning tasks at the pair-level of nodes. I talked about side effect prediction and we discussed recommender systems. I would want to discuss the sub-graph level machine-learning task now. Moreover [here] is one that is relatively current that we all use on a daily basis. It has to do with predicting traffic.

For instance, if you open Google Maps today and tell it that you want to drive from Stanford to Berkeley, it will tell you how long it would take you to get there and what time you should arrive.

I am not sure if you knew this, but in the end, graph machine learning is utilized to construct these journey time forecasts. The graph is created with nodes that represent individual road segments and edges that capture connectedness between those segments. Then, our graph neural network approach is trained to forecast the estimate that time of arrival or travel time based on the conditions and traffic patterns on each of the road segments as well as the path between the source and the destination of the voyage. It has been revealed that Google Maps [27] actually uses this graph-based strategy in production. As a result, anytime you ask for directions, a graph machine learning approach actually tells you when you will arrive at a specific area.

Finally, I would want to discuss some intriguing and useful applications of graph-level machine learning tasks. One is about drug discovery. In fact, new medications and antibiotics have been discovered using graph-based machine learning [3]. Antibiotics are tiny molecular graphs, and we can visualize molecules as graphs with atoms as nodes and chemical bonds as edges. Therefore, a graph can be used to depict each molecule. However, we also have these collections of countless molecular units. Which chemicals might have therapeutic effects is the question.

In other words, which compounds ought to be given priority so that scientists can test them in the lab to confirm their therapeutic impact? In fact, a team at MIT [24] employed a graph-based deep learning strategy for antibiotic discovery, classifying various chemicals and predicting potential molecules from a pool of billions of candidates using a graph neural network. Then, in the lab, these predictions would have been further confirmed. Recently, a very intriguing, ground-breaking work on the use of graph-based approaches to uncover new medications and novel therapeutic applications for various types of molecules was discovered [1].

To continue our discussion of drug discovery, let us consider graph generation as a method for finding novel chemicals that have never been synthesized or taken into consideration. In addition, this is incredibly helpful because it enables us to create new molecules and structures in a variety of focused ways. For instance, we could say, "Generate novel compounds that are non-toxic, high solubility, and high drug resemblance." Therefore, we can generate now molecules as graphs in a targeted method.

A second application involves modifying current molecules to have a desired attribute. The use case in this instance is that you have a little portion of the molecule that, for instance, has a specific therapeutic impact. Now we want to finish the remaining portions of the molecular scaffold to enhance a specific attribute. Solubility and these kinds of deep network generative models, for instance, can be applied to activities like the creation and optimization of molecules.

A realistic, physics-based simulation [30], [11], [17] is the final graph-level challenge I want to discuss. In this situation, we essentially have a choice of materials. We may construct a graph on top of this set of particles that we use to represent the material that shows how the particles interact with one another. Predicting how this graph will change in the future is now the fundamental task for machine learning. Moreover, this enables us to forecast the deformation of this material. So, let me explain how to achieve this. In order to accomplish this, we iterate the next strategy. We take the substance and turn it into a collection of particles. We created the proximity graph based on the proximities and interactions between the particles. Now that we have this proximity graph, we can use graph machine learning—a graph neural network—to forecast future positions and velocities of the particles based on their current attributes, which include their positions and speeds. We can now move and evolve the particles to their new positions based on this prediction, and then we return to the first phase where we generate a new graph based on the new proximities, predict the new positions, move the particles, and repeat this process. In addition, this makes it possible to run physics-based simulations very quickly and accurately.

By drawing an increasing number of youthful users, live-streaming platforms [19] have recently experienced substantial growth in popularity and have emerged as one of the most promising methods of online commerce. Live-streaming platforms, like more conventional online retail sites like Taobao, are also plagued by malevolent online fraud where many of the transactions are not real. On platforms for live streaming, the anti-fraud methods currently in use are not suitable for identifying fraudulent transactions. This is mostly due to the distinctive sort of heterogeneous live-streaming networks that live-streaming platforms employ to connect several heterogeneous types of nodes, including users, livestreamers, and products, using a variety of different types of edges and edge features.

In their paper [25], authors offer a novel method for detecting live streaming fraud based on a heterogeneous graph neural network (called LIFE). With the help of a certain live-streaming platform's viewers, streamers, and other heterogeneous information, LIFE creates an inventive heterogeneous graph learning model. Additionally, our LIFE framework uses a label propagation technique to manage the small number of identified fraudulent transactions needed for model training. The suggested method outperforms the baseline models in terms of live-streaming platform fraud detection efficacy, according to extensive experimental data on a large-scale Taobao live-streaming platform. Additionally, we carry out a case study to demonstrate that the suggested technique.

These were some instances of graph-level problems and significant applications of graph machine learning to a variety of disciplines, including, but not limited to, the sciences, industry, and numerous consumer goods.

III. Representations

I want to discuss the selection of graph representation in this section. What then make up a network or a graph? There are two different categories of things in a network. First, there are the actual objects or things themselves, which are referred to as nodes and vertices, and then there are the connections between them, which are referred to as links or edges. In addition, after that, the entire system or domain is referred to as a network or a graph.

The letter capital N or capital V is typically used to represent nodes, while the letter capital E is typically used to represent edges, resulting in the graph G being made up of a set of nodes (N) and a set of edges (E). The key benefit of graphs is that they speak a common language. This means that I can connect actors, for instance, based on the films they appeared in, or I can connect people based on their relationships with one another, or I can connect molecules, such as proteins, based on how those proteins interact with one another.

This means that the same machine-learning algorithm will be able to make predictions regardless of whether these nodes correspond to actors, correspond to people, or they correspond to molecules like proteins if I look at the structure of this network and what the underlying mathematical representation is in each of these cases. Of course, picking a suitable graph representation is crucial. We can connect people who work with each other, for instance, if you have a group of people, and this will create a professional network. The same group of people can also be connected based on their musical preferences and interactions, however doing so will result in the creation of a social music network. Alternatively, if we have a group of scientific articles, we can connect them based on citations, or which paper cites which other paper. For instance, the underlying network's and the underlying representations' quality might be significantly lower if we connected them based on whether they share a word in the title.

Therefore, it's crucial to choose carefully what the nodes and links are. Therefore, whenever we are given a data collection, we must decide how to create the underlying graph, including which nodes will represent the objects of interest, how those nodes will be related to one another, and which edges will connect them. Our ability to properly use networks will be determined by the selection of the appropriate network representation for a given topic or problem. In some instances, there will be a distinct, unambiguous way to describe this issue or topic as a graph; yet, in other instances, this representation may not even be distinct. The topics we will be able to research and the kinds of predictions we will be able to make will depend on how we assign linkages between the items. I will now go over some principles and several sorts of graphs that we can construct from data in order to illustrate some examples of the design decisions we must make when co-creating them.

To start, I will make a distinction between directed [35] and undirected graphs [7]. Undirected graphs have links that are undirected (bi-directional), which makes them useful for modeling symmetric or reciprocal relationships like cooperation, friendship, interaction between proteins, and so forth. Directed graphs, on the other hand, have links that are directed (one direction), where each link has a direction, a source, and a destination indicated by an arrow. Phone calls, financial transactions, following on Twitter, and other real-world instances of links with a source and a destination include these forms of links.

For undirected graphs, we can discuss the idea of a node degree, which is the second form of graph that we will discuss. The number of edges that are immediately close to a specific node is known as the node degree. The average node degree is just that—the average of all the nodes in the network's degrees. In addition, if you do the math, it comes out to be twice as many edges as there are nodes in the network. The reason for the number 2 is that each edge is counted twice when calculating the nodes' degrees, correct. Because both ends attach to the same node, having a self-edge or self-loop adds a degree of two rather than one to the node.

We distinguish between in-degree and out-degree in directed networks; in-degree is the quantity of edges that point in the direction of the node. Another extremely common sort of graph structure that is used frequently and is highly natural in several disciplines is known as a bipartite graph. Additionally, a bipartite graph is a network of nodes that often consists of two different types of nodes and in which nodes only communicate with other nodes of the same type.

A bipartite graph, for instance, is one in which the nodes can be divided into two divisions, and in which the edges only move from the left partition to the right partition and not inside the same partition. Bipartite graphs are examples that naturally exist, such as links between writers of scientific articles and the papers they published, actors and the movies they appeared in, users and the movies they rated or viewed, and so on. For instance, a bipartite graph representing people purchasing goods would have a set of customers, a set of goods, and a link connecting the client to the good she bought.

Following the definition of a bipartite network, we can now describe the concepts of a folded or projected network, allowing us to construct networks for author cooperation or movie co-rating, among other things. Furthermore, the concept is as follows: if I have a bipartite graph, I can project it to either the left or the right side. And then- and when I project it, I essentially only utilize the nodes from one side in my projection graph, and I connect the nodes by saying that if two nodes have at least one neighbor in common, I will make a connection between them. In addition, as I mentioned, bipartite or multipartite graphs are highly common if you have numerous sorts of edges. This is particularly true if you have two different categories of nodes, such as users and items, users and movies, authors and papers, and so on. How we represent graphs is another intriguing aspect of them; and how do we represent graphs is an intriguing subject.

A graph can be represented using an adjacency matrix, for example. In other words, if we have an undirected graph with, say, four end nodes, for example see Figure 2, we will generate a square matrix with this matrix being binary.

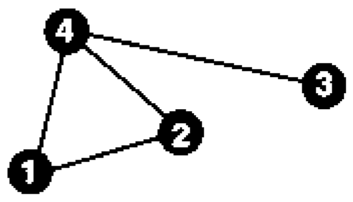

Fig. 2. Undirected graph

The only values accepted are 0 and 1. In essence, if nodes i and j are connected, an element of matrix i, j will be set to 1; otherwise, it will be set to 0. For instance, since 1 and 2 are linked, there is a 1 at entry 1, row 1, column 2. And also, because 2 is connected to 1 at row 2, column 1, we also have a 1. As a result, adjacency matrices of undirected graphs are symmetric by nature, see Figure 3.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Fig. 3. Adjacency matrix for an undirected graph.

Figure 4 shows is a directed graph.

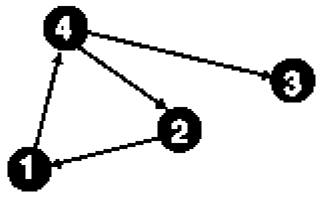

Fig. 4. A directed graph.

The matrix will not be symmetric if the graph is directed since 2 links to 1. There is a 0 because even though we have a 1, it does not connect to a 2, see Figure 5.

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Fig. 5. Adjacency matrix for a directed graph.

In a similar vein, we can consider node degrees to be nothing more complicated than a summing across a given row or a given column of the graph's adjacency matrix. So instead of kind of trying to figure out how many edges are adjacent here, we can simply go ahead and add up- basically, count how many other nodes this particular node is related to. This is for undirected graphs. For directed graphs, in and out degrees will be sums over the columns and sums over the rows of the graph adjacency matrix.

The fact is that real-world networks are incredibly sparse. This implies that if you were to examine the adjacency matrix, or series on adjacency matrix, of a real-world network, where for every, row i, column j, (we put a dot if there is an edge and otherwise the cell is empty) you would obtain sparse matrices where, where there are significant portions of the matrix that are empty. As a result of their great sparsity, these matrices' characteristics are affected in a significant way.

Let me give you an illustration. The maximum degree of a node, or the number of connections a node has, is n minus one if your network contains n nodes, so you can theoretically link to any other node in the system. In other words, if you consider yourself a person and consider the human social network, the total number of friends you may possibly have is every other human in the entire planet. Nobody, however, has seven billion friends, is that right? Our social circle is much, much smaller.

Therefore, let us say that the human social network is exceedingly sparse. It turns out that many other different types of networks, such as power grids, Internet connections, scientific collaborations, email graphs, and so on and so forth, are also extremely sparse. They have an average degree of about 10 or even up to 100, you know. So what is the result? The underlying adjacency matrices are hence very sparse. As a result, we have always represented the matrix as a sparse matrix rather than a dense matrix.

There are two other ways to display graphs. One is to simply represent it as a list of edges, or an edge list. Because we can easily describe it as a two-dimensional matrix, this representation is extremely common in deep learning systems. The issue with this model is that it is exceedingly difficult to perform any type of graph modification or analysis because, in this instance, even calculating a node's degree is not straightforward. The idea of an adjacency list is a much, much, better representation for a graph's examination and manipulation. Adjacency lists are beneficial because they are simpler to use in big and sparse networks. In addition, an adjacency list only enables us to rapidly access all of a given node's neighbors.

You may think of it as simply storing a list of a node's neighbors for each node. Consequently, a list of nodes to which a particular node is connected. In the case of an undirected graph, neighbors could be stored. If the graph is connected, it is possible to record both the outgoing and incoming neighbors based on the edge's orientation.

The final significant point I want to make is that these graphs can, of course, have properties linked to them. As a result, attributes or properties can be connected to node addresses as well as complete graphs. Therefore, for instance, a weight could exist on an edge.

How strong is the relationship? It might be able to have my ranking. It may possess a type. It may be clear whether a connection is built on friendship or on hostility, complete mistrust, or anything similar. Moreover, edges can have a variety of various types of attributes, such as the duration of a phone call. If the nodes represent people, the properties they may have include age, gender, interests, location, and so forth. If a node is a chemical, its chemical mass, chemical formula, and other chemical qualities might be represented as the node's attributes.

Additionally, complete graphs may include traits or features dependent on the characteristics of the underlying object that the graphical structure is describing.

As a result, the graphs you will be evaluating will have both their associated properties and their topological nodes and edges. As I said, some of these features can also be directly expressed in the adjacency matrix. So the adjacency matrix may easily capture, for instance, edge attributes like weights? We may now have adjacency matrices with real values instead of binary ones, where the strength of the connection simply relates to the value of that element. As a result, the value for the link between two and four is four, whereas the relationship between one and three has only a 0.5 weight and is therefore weaker.

As an additional essential point, we should consider the possibility of self-looping nodes while designing graphs. For instance, node four in this instance has a self-loop, and as a result, its degree is now equal to three. Self-loops match the items in the adjacency matrix's diagonal. In addition, in some circumstances, we might even construct a multi-graph in which we permit several edges to exist between a pair of nodes.

A multi-graph can sometimes be thought of as a weighted graph where the entry on the matrix counts the number of edges, but there are other times when you want to represent each edge independently because they may have various characteristics and traits. In nature, multi-graphs and self-loops both happen rather frequently. For instance, if you consider phone call transactions, there may be several transactions taking place between a pair of nodes, and we can precisely model this as a multi-graph.

Since we are dealing with graphs, I also want to discuss the idea of connectedness, or more specifically, whether a graph is linked or disconnected. A graph is said to be linked if any pair of nodes in the graph can be connected by a path that runs along its edges.

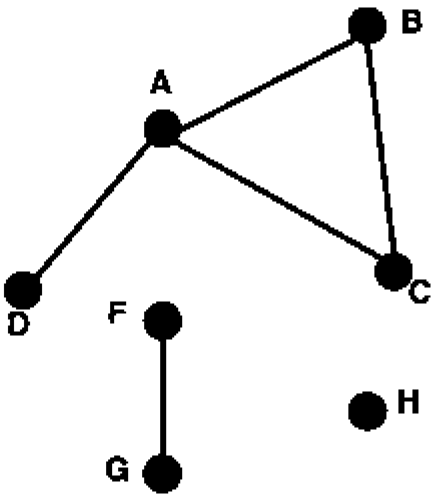

Fig. 6. Disconnected graph: made up by two or more connected components

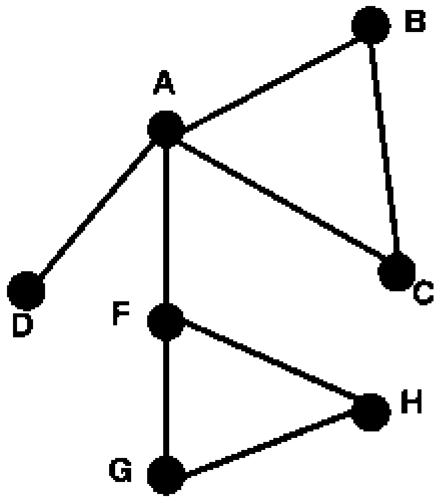

Fig. 7. Connected graph: any two vertices can be joined by a path

As an illustration, this particular graph has three connected components (see Figure 6) while the other graph does not (see Figure 7), making it the connected graph. This has three connected components: a single connected component, a second connected component, and a third connected component, the isolated node h.

This is the idea of connectivity for undirected graphs, and what is interesting about it is that when we disconnect a graph and examine the structure of the underlying adjacency matrix, we will see these block diagonal structures. In essence, if a graph is made up of two components, then we will have block diagonal structures where the edges only connect the nodes inside each component.

Directed graphs can also be used to broaden the concept of connection. Strong connectivity and poor connectivity are the two forms of connectivity being discussed. Simply put, a graph that is linked if we ignore the direction of the edges is referred to as a weakly connected directed graph. A strongly connected graph or a graph is connected if there is a directed path between every pair of its nodes. This means that if the network is strongly connected, a directed path must exist from, for instance, node A to node B and from node B back to node A.

This also means that we may discuss the idea of strongly connected components [20], which are collections of nodes in a network where each node can contact another node in the collection via a directed path. Because they are on a cycle, for instance, nodes A, B, and C in this instance constitute a tightly connected component. Therefore, we may visit any other node from any node.

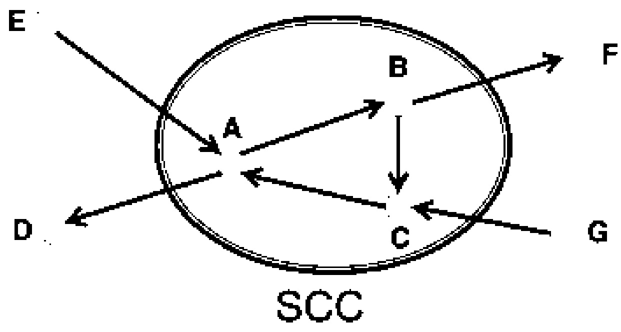

Fig. 8. Nodes A, B, and C in this instance constitute a tightly connected component.

In this example shown on Figure 9, a directed graph with two strongly connected components, again with two cycles on three nodes.

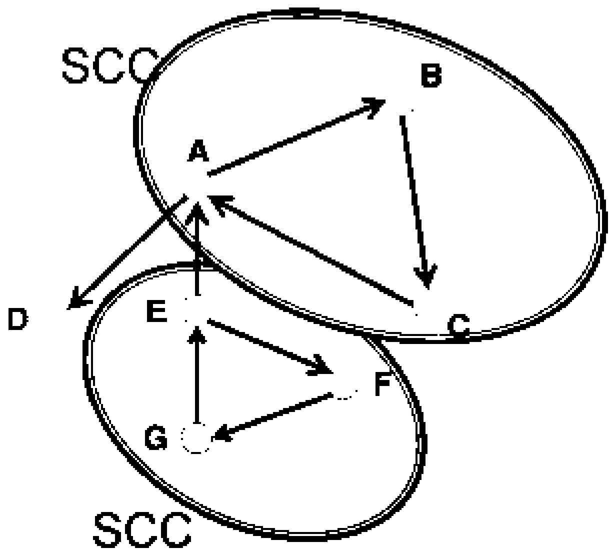

Fig. 9. A directed graph with two strongly connected components.

Thus, the topic of graph representations and methods for making them from actual data is now complete.

IV. CONCLUSION

In the first part of this paper, we discussed machine learning with graphs and its different use cases. Machine-learning prediction challenges at the node level, edge level, and graph level were discussed. The selection of a graph representation was then discussed in terms of directed and undirected graphs, bipartite graphs, weighted and unweighted graphs, adjacency matrices, and some definitions from graph theory, such as the connectivity of graphs, weak connectivity, and strong connectivity, as well as the concept of node degree.

REFERENCES

[1] Al Mahi, Naim, et al. "Connectivity map analysis of a single-cell RNA-sequencing-derived transcriptional signature of mTOR signaling." International journal of molecular sciences 22.9 (2021): 4371.
[2] Asif, Nurul A., et al. "Graph neural network: A comprehensive review on non-euclidean space." IEEE Access 9 (2021): 60588-60606.
[3] Atance, Sara Romeo, et al. "De novo drug design using reinforcement learning with graph-based deep generative models." Journal of Chemical Information and Modeling 62.20 (2022): 4863-4872.
[4] Barros, Claudio DT, et al. "A survey on embedding dynamic graphs." ACM Computing Surveys (CSUR) 55.1 (2021): 1-37.
[5] Cai, Weiwei, et al. "Voxel-based three-view hybrid parallel network for 3D object classification." Displays 69 (2021): 102076.
[6] Cao, Jiangxia, et al. "Bipartite graph embedding via mutual information maximization." Proceedings of the 14th ACM International Conference on Web Search and Data Mining. 2021.
[7] Dalirrooyfard, Mina, Ray Li, and Virginia Vassilevska Williams. "Hardness of approximate diameter: Now for undirected graphs." 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2022.
[8] David, Alessia, et al. "The AlphaFold Database of Protein Structures: A Biologist's Guide." Journal of Molecular Biology 434.2 (2022): 167336.
[9] Ding, Yijie, Jijun Tang, and Fei Guo. "Identification of drug-target interactions via multi-view graph regularized link propagation model." Neurocomputing 461 (2021): 618-631.
[10] Dong, Xiaowen, et al. "Learning graphs from data: A signal representation perspective." IEEE Signal Processing Magazine 36.3 (2019): 44-63.
[11] Gao, Han, Matthew J. Zahr, and Jian-Xun Wang. "Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems." Computer Methods in Applied Mechanics and Engineering 390 (2022): 114502.
[12] Ghorbani, Modjtaba, et al. "On the eigenvalue and energy of extended adjacency matrix." Applied Mathematics and Computation 397 (2021): 125939.
[13] Granskog, Jonathan, et al. "Neural scene graph rendering." ACM Transactions on Graphics (TOG) 40.4 (2021): 1-11.
[14] Gutierrez, Claudio, and Juan F. Sequeda. "Knowledge graphs." Communications of the ACM 64.3 (2021): 96-104.
[15] Hanspers, Kristina, et al. "Ten simple rules for creating reusable pathway models for computational analysis and visualization." PLoS Computational Biology 17.8 (2021): e1009226.
[16] Hoel, Robert William, Ryan M. Giddings Connolly, and Paul Y. Takahashi. "Polypharmacy management in older patients." Mayo Clinic Proceedings. Vol. 96. No. 1. Elsevier, 2021.
[17] Iiyama, Yutaro, et al. "Distance-weighted graph neural networks on FPGAs for real-time particle reconstruction in high energy physics." Frontiers in big Data 3 (2021): 598927.
[18] Kasanishi, Tetsu, Xueting Wang, and Toshihiko Yamasaki. "Edge-Level Explanations for Graph Neural Networks by Extending Explainability Methods for Convolutional Neural Networks." 2021 IEEE International Symposium on Multimedia (ISM). IEEE, 2021.
[19] Khan, Koffka, and Wayne Goodridge. "QoE in DASH." International Journal of Advanced Networking and Applications 9.4 (2018): 3515-3522.
[20] Kumar, Sudhakar, et al. "Evaluation of automatic parallelization algorithms to minimize speculative parallelism overheads: An experiment." Journal of Discrete Mathematical Sciences and Cryptography 24.5 (2021): 1517-1528.
[21] Li, Jianxin, et al. "Higher-order attribute-enhancing heterogeneous graph neural networks." IEEE Transactions on Knowledge and Data Engineering 35.1 (2021): 560-574.
[22] Li, Michelle M., Kexin Huang, and Marinka Zitnik. "Graph representation learning in biomedicine and healthcare." Nature Biomedical Engineering (2022): 1-17.
[23] Li, Mufei, et al. "Dgl-lifesci: An open-source toolkit for deep learning on graphs in life science." ACS omega 6.41 (2021): 27233-27238.
[24] Li, Pengyong, et al. "An effective self-supervised framework for learning expressive molecular global representations to drug discovery." Briefings in Bioinformatics 22.6 (2021): bbab109.
[25] Li, Zhao, et al. "Live-Streaming Fraud Detection: A Heterogeneous Graph Neural Network Approach." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.
[26] Murali, Ramachandran, and Mark I. Greene. "Structure based antibody-like peptidomimetics." Pharmaceuticals 5.2 (2012): 209-235.
[27] Naiudomthum, Supiya, Ekbordin Winijkul, and Sunicha Sirisubtawee. "Near Real-Time Spatial and Temporal Distribution of Traffic Emissions in Bangkok Using Google Maps Application Program Interface." Atmosphere 13.11 (2022): 1803.
[28] Ovchinnikov, Sergey, and Po-Ssu Huang. "Structure-based protein design with deep learning." Current opinion in chemical biology 65 (2021): 136-144.
[29] Rosinol, Antoni, et al. "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs." The International Journal of Robotics Research 40.12-14 (2021): 1510-1546.
[30] Schuetz, Martin JA, J. Kyle Brubaker, and Helmut G. Katzgraber. "Combinatorial optimization with physics-inspired graph neural networks." Nature Machine Intelligence 4.4 (2022): 367-377.
[31] Shi, Yingying, et al. "Exploring the dynamics of low-carbon technology diffusion among enterprises: An evolutionary game model on a two-level heterogeneous social network." Energy Economics 101 (2021): 105399.
[32] Tay, Christina. "Econometric Models to Estimate the Impact of Social Media Platforms On E-commerce: Pre-and Post-COVID." 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2021.
[33] Wang, Zichen, et al. "Lm-gvp: an extensible sequence and structure informed deep learning framework for protein property prediction." Scientific reports 12.1 (2022): 1-12.
[34] Wu, Lingfei, et al. "Graph neural networks." Graph Neural Networks: Foundations, Frontiers, and Applications. Springer, Singapore, 2022. 27-37.
[35] Zhang, Xitong, et al. "Magnet: A neural network for directed graphs." Advances in Neural Information Processing Systems 34 (2021): 27003-27015.