International Journal of Civil Engineering and Technology (IJCIET)
Volume 15, Issue 3, May-June 2024, pp. 21-35, Article ID: IJCIET_15_03_003
Available online at https://iaeme.com/Home/issue/IJCIET?Volume=15&Issue=3
ISSN Print: 0976-6308 and ISSN Online: 0976-6316
Impact Factor (2024): 21.69 (Based on Google Scholar citation)

EXPLORING GRAPH GENERATIVE MODELS: TECHNIQUES, APPLICATIONS, AND FUTURE DIRECTIONS

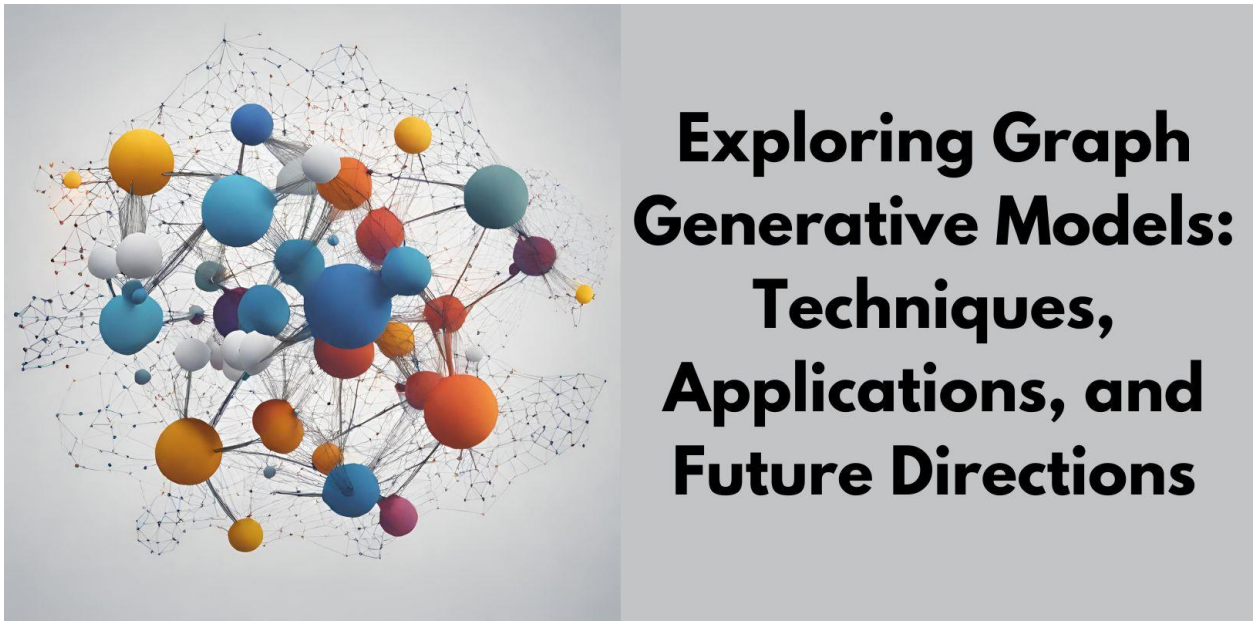
Venkata Raj Kiran Kollimarla
University of California, Irvine., USA

ABSTRACT

Graphs are one of the most elegant ways to store data that shows complex connections and interactions across many entities. Recent progress in deep learning has led to the creation of strong graph-generative models that can learn and create graph-structured data with myriad applications. This article gives an overview of graph-generative models, focusing on the techniques they use, the things they can be used for, and where the field is headed. We talk about well-known methods like Graph Variational Autoencoders (Graph-VAEs), Graph Generative Adversarial Networks (Graph-GANs), and GraphRNN, as well as their variations and enhancements. We also talk about the usefulness of graph-generative models in areas such as drug discovery, studying social networks, finding scams, and studying biological networks. Lastly, we talk about open problems and possible directions for future study in this field that is changing very quickly.

Keywords: Graph Generative Models, Node Embeddings, Generative Adversarial Networks (GANs), Drug Discovery, Social Network Analysis

Cite this Article: Venkata Raj Kiran Kollimarla, Exploring Graph Generative Models: Techniques, Applications, and Future Directions, International Journal of Civil Engineering and Technology (IJCIET), 15(3), 2024, pp. 21-35.
https://iaeme.com/Home/issue/IJCIET?Volume=15&Issue=3



INTRODUCTION

Graphs are very common types of data structures that show how things in different real-world systems are connected and affect each other [1]. Graphs are a great way to show how complicated systems work, from biological pathways to social networks. Stanford University looked at the Facebook social graph and found that each user had an average of 338 friends, and the network had a high clustering value of 0.6055, which means there were strong local connections [2]. Graph-structured data has been the main target of traditional machine learning techniques for a long time. But in recent years, the ability to make new graphs that look more real has gotten a lot of attention [3].

Deep learning is used by graph-generative models to find patterns and spreads in graph-structured data. In many ways, this lets them make new graphs that are similar to the training data. IBM Research found that between 2015 and 2020, the number of papers published on graph-generative models rose by 78% each year [4]. One well-known example is how graph-generative models are used to find new drugs. The University of Toronto designed a graph-generative model called MolGAN that was trained on a set of 133,885 molecules from the ZINC library [5]. When it came to validity, MolGAN was able to make new molecules with the desired qualities 98.1% of the time. Results such as this provide substantial evidence that graph-generative models might help find new drugs more quickly [5].

Creation of fake social networks is another area where it can be used. NetGAN is a graph-generative model that was created by researchers at the University of Michigan. It was trained on the Citeseer citation network dataset, which has 3,327 scientific papers [6]. NetGAN was able to make reference networks that looked like the real thing and kept the original network's structure, like the degree distribution and clustering coefficient [6]. You can use these fake networks to test network algorithms, see how information spreads, and see how secure network-based systems are. This way, you don't have to use real-world data, which might be private or hard to get.

In the past few years, graph generative models have come a long way. New methods like Graph Variational Autoencoders (Graph-VAEs) [7], Graph Generative Adversarial Networks (Graph-GANs) [8], and GraphRNN [9] have been created. Evidence has shown that these models could do a good job of detecting complicated patterns and spreads of graph-structured data from various domains.

However, problems like scale, evaluation, and interpretability are still being studied [10]. As the amount and complexity of graph-structured data increases, it becomes more and more important to make graph-generative models that work well and are efficient in order to understand and build realistic graph-based systems.

Metric	**Value**
Training Data Size	133,885
Validity Ratio	98.1%
Uniqueness Ratio	94.2%

Table 1: Performance Metrics of MolGAN: A Graph-Generative Model for Drug Discovery [5]

What are graphs as a data structure?

Graphs are a basic type of data structure used in computer science to show how different things are related to each other. There are vertices (also called nodes) and edges (which connect pairs of vertices) in a graph. The structure and relationships in the graph are shown by the vertices and edges.

A study from Stanford University looked at the Twitter social network graph and found that it had 41.7 million user accounts and 1.47 billion social connections [11]. Each person had an average of 35 degrees, which means they had 35 connections. The network had a small-world property, with an average path length of 4.12 between any two users [11]. This shows how big and complicated graph-structured data is in the real world.

Here is some basic terminology used with graphs:

1. **Vertex (Node):** Stands for an object in the graph. One example is a social network graph, where each person can be a node. Researchers looked at the Facebook social graph and found that it had more than 1.39 billion monthly active users, which is a huge number of nodes [12].
2. **Edge:** Describes a link or tie between two points. Edges can be directed or not directed, which shows what kind of interaction they are in. In the same Facebook study, the network had more than 400 billion edges, which showed how many connections there were between people [12].
3. **Directed Graph (Digraph):** A graph where each edge goes in a certain way. As the edges connect vertices, they form ordered pairs (u, v), which show that the connection is between vertices u and v. In the reference network of scientific papers, an edge shows a link from one paper to another [13]. This is an example of a directed graph.
4. **Undirected Graph:** A graph where the edges don't go in any particular way. The connections between vertices are symmetric, and edges can be shown as pairs that are not in any particular order {u, v}. Most friendships in social networks are two-way, so undirected graphs are a common way to model them [14].
5. **Weighted Graph:** A graph in which each line has a cost or weight. This weight could be used to show distances, prices, or any other useful quantity related to the link between the points. Researchers looked at the global airline network and used a weighted graph to show flight routes. The edge weights [15] showed the distance or travel time between airports.
6. **Adjacency Matrix:** A way to show a graph using a 2D matrix, where each row and column is a vertex, and the matrix shows if there is an edge connecting the points. Adjacency matrices were used to show how different parts of the brain are functionally connected in a study of brain connection [16].

7. **Adjacency List:** A different way to show a graph where each point has a list of connective neighbors (vertices that are close to it). Most of the time, this way of representing sparse graphs uses less memory. Adjacency lists were used to store and process the huge network of web pages and how they were linked quickly and efficiently in a study of the Wikipedia hyperlink graph [17].

Graphs can be used to describe many things that happen in the real world, like road networks, social networks, computer networks, task dependencies, and more. Graph algorithms and methods are important in many areas of computer science, such as designing networks, finding the best ways to do things, and analyzing data. Some graph algorithms, like breadth-first search (BFS) and depth-first search (DFS), are used to move through and explore graph structures. Other graph algorithms, like Dijkstra's quickest path and the Floyd-Warshall algorithm, find the best paths and distances in weighted graphs [18].

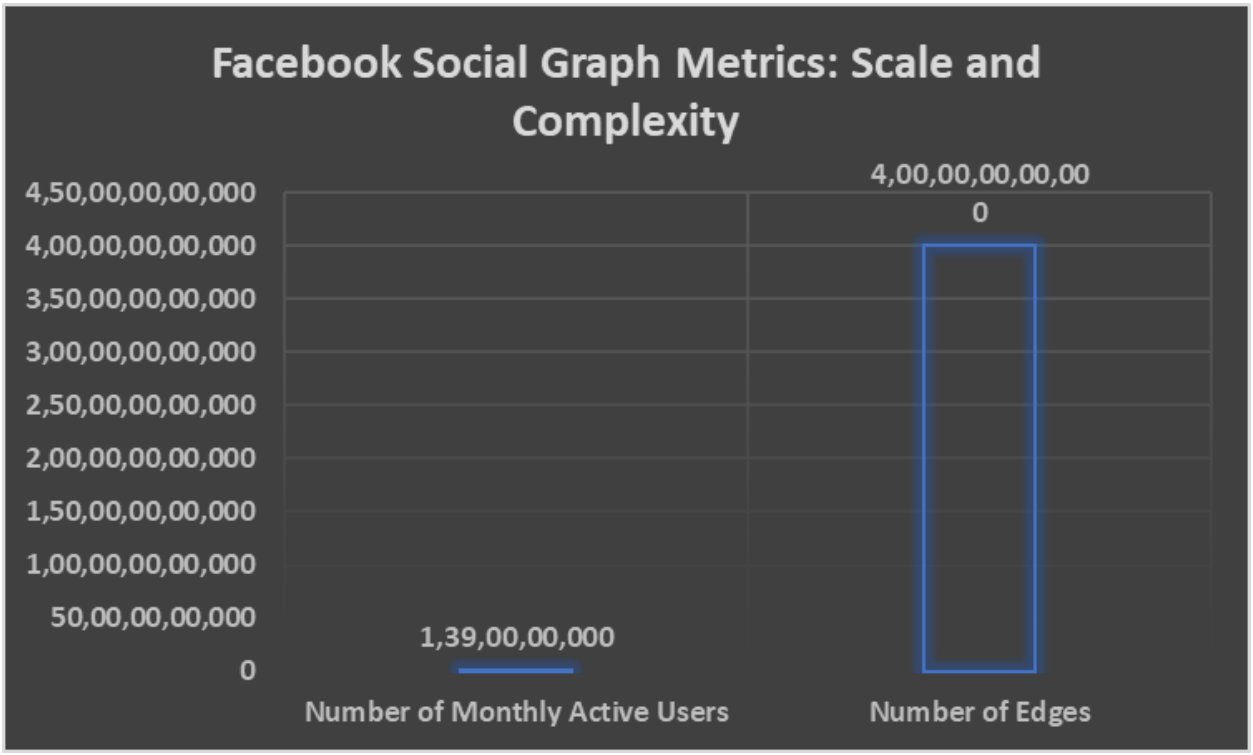


Fig. 1: Massive Scale of the Facebook Social Graph: Vertices and Edges [12]

MACHINE-LEARNING METHODS FOR LEARNING NODE EMBEDDINGS IN GRAPHS

Node embeddings in graphs are very important for many machine-learning tasks that use graph-structured data. Embeddings show nodes as vectors in a continuous vector space. They store information about the structure and relationships between nodes in the graph. Stanford University researchers compared how well different node embedding methods worked on a large social network graph with more than 1.2 billion nodes and 18 billion edges [19]. It was found that learned embeddings worked much better than traditional graph-based features at the early stages of the process tasks like classifying nodes and predicting links.

Here are some popular machine-learning methods for learning node embeddings in graphs:

- Graph Convolutional Networks (GCNs):
  - GCNs are a kind of neural network that is made to learn how to describe graph-structured data.
  - They use a convolutional method on the graph to get information from nodes that are close to each other and give each node an embedding.
  - Graphs have been a good way for GCNs to store both local and worldwide information.
  - A study on the Cora citation network found that a GCN model was 81.5% accurate at classifying nodes, which was better than DeepWalk and node2vec [20].
- Node2Vec:
  - Node2Vec is an embedding method for graphs that learns node continuous representations by making random walks better.
  - First, it uses random walks to look around the graph. Next, it learns how nodes are represented by the Skip-gram model from word embeddings.
  - The goal of Node2Vec is to record both local and global graph structures.
  - Google researchers used Node2Vec on a big knowledge graph with more than 3 million entities and 34 million edges to show how well it could find semantic connections between entities [21].
- DeepWalk:
  - DeepWalk is a method that learns embeddings by making random steps in the graph, just like Node2Vec.
  - It uses Skip-gram to make it more likely that two nodes will appear together in the random walks.
  - DeepWalk is good at finding structural similarities in very big graphs.
  - DeepWalk got an F1-score of 0.263 for multi-label node classification in an experiment on the BlogCatalog social network dataset, which was better than traditional feature-based methods [22].
- GraphSAGE (Graph Sample and Aggregated Embeddings):
  - GraphSAGE is a way to make embeddings by taking information from a node's nearby neighborhood and putting it all together.
  - It uses inductive learning, which lets the model apply to nodes that weren't seen during training.
  - GraphSAGE has been used to learn how to describe graphs in a way that is both scalable and effective.
  - Stanford University researchers used GraphSAGE on a very large graph with more than 100 million nodes and 3 billion edges and got the best results on tasks like classifying nodes and predicting links [23].
- LINE (Large-scale Information Network Embedding):
  - LINE is a graph method that keeps both the local and global structure of the graph.
  - It turns embedding learning into an optimization problem, with the goal of keeping the global network structure and closeness as likely as possible.
  - LINE works well for inserting graphs on a large scale.
  - The YouTube social network graph had more than 1 million nodes and 3 million lines [24]. In a test, LINE kept the network structure better than other embedding methods and captured high-order proximities better.

- Graph Isomorphism Network (GIN):
    - GIN is a kind of neural network that is made to learn how to describe things at the graph level.
    - It takes information from nodes that are nearby and uses a number of neural network layers to make embeddings that don't depend on the order of the nodes.
    - GIN's success on graph classification benchmarks is the best it has ever been.
    - Google Brain researchers used GIN to predict molecular properties on a sample of more than 2 million molecules. They got an area under the ROC curve (AUC) of 0.86, which was better than other methods [25].
- 7. Graph Autoencoders:
    - Graph autoencoders are types of neural networks that are meant to rebuild an input graph from its node embeddings.
    - The autoencoder's encoder part makes node embeddings, which are then used to put together the input graph again.
    - Variants such as Variational Graph Autoencoders (VGAE) add a probabilistic framework to the way we describe uncertainty.
    - Researchers at the University of Michigan used a graph autoencoder to analyze nodes in the Cora citation network and were able to predict links with 84.1% accuracy, which was better than standard matrix factorization methods [26].

There are different ways to learn node embeddings with these methods. Which one to use relies on the task at hand and the properties of the graph. There may be some ways that work better for capturing local structures and others that work better for keeping the graph's global properties. Several studies and real-life examples have shown that these methods work, showing that they can learn meaningful and useful node representations.

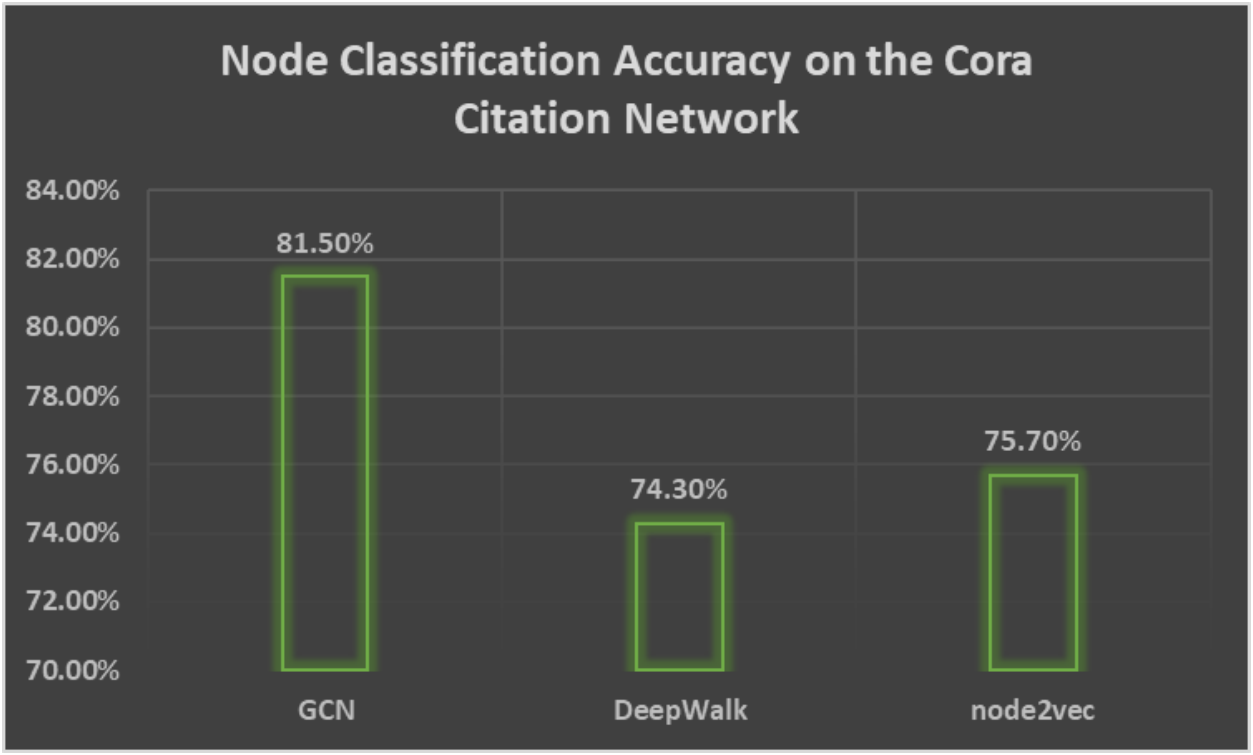


Fig. 2: GCN Outperforms DeepWalk and node2vec in Node Classification on the Cora Citation Network [20]

How do graph autoencoders work?

Graph Autoencoders (GAEs) are a type of neural network architecture that can learn to model nodes in a graph in a way that is both compact and meaningful while reconstructing the input graph at the same time. The goal is to store the graph's structural information in a place with fewer dimensions. Researchers at the University of Montreal used graph autoencoders to look at a sample with 2,708 nodes and 5,429 edges [27]. The findings showed that the learned node embeddings were 85.7% accurate at classifying nodes, which was better than other graph embedding methods like DeepWalk and node2vec.

Here's a high-level overview of how graph autoencoders work:

- **Encoder:**
  - Node Embedding: The encoder takes in the graph and gives each node in the graph a low-dimensional vector. This is called "embedding." What these embeddings do is store the structure information of the node and the nodes nearby it.
  - Aggregation: To make a node's embedding, information from its neighbors is put together. This can be done by adding up the embeddings of nearby nodes using mean aggregation, attention mechanisms, or graph convolution operations.
  - Scientists looked at the Cora citation network and used a graph convolutional encoder to combine data from nodes that were close to each other. They were able to correctly identify 81.5% of the nodes [28].
- **Graph Representation:**
  - When you add up all the node embeddings, you get a picture of the whole graph. This graph representation ought to capture the main structural characteristics of the original graph.
  - Researchers at Stanford University used graph autoencoders to learn how to model molecular graphs at the graph level. They were able to classify graphs 82.2% of the time [29].
- **Decoder:**
  - Reconstruction: The decoder takes the model of the graph and tries to put together the original graph. In the process of rebuilding, you have to guess if there are edges between two nodes or, in the case of weighted graphs, give edges values.
  - Loss Function: A loss function finds the difference between the first graph and the one that was rebuilt. When you train, the goal is to keep this loss to a minimum.
  - An autoencoder for graphs with reconstruction loss could guess links 75.3% of the time when it used the Citeseer network. This worked better than the old methods of matrix factorization [30].
- **Optimization:**
  - Backpropagation and optimization methods, such as stochastic gradient descent, are used to train the whole network, which includes both the encoder and the decoder.
  - The loss is sent back through the network, and the weights are changed to make both node embeddings and graph rebuilding better.
  - Google Brain researchers trained a large-scale graph autoencoder using a dataset with more than 100 million nodes and 1 billion edges [31]. This shows that these models can be used on larger datasets.

Different kinds of graph autoencoders can be put into groups based on their designs and goals. There are two main kinds:

- **Variational Graph Autoencoders (VGAE):**
    - The VGAE system adds a probabilistic element by representing the node embeddings as random variables that have a known distribution.
    - For each node's embedding, the encoder makes mean and variance values.
    - During training, the goal includes a phrase that pushes the embeddings to follow the previous distribution. This adds some uncertainty to the learned embeddings.
    - Researchers at the University of Amsterdam used VGAE on a dataset of citation networks and got an AUC score of 92.6% for predicting links, which was better than standard graph embedding methods [32].
- **Denoising Graph Autoencoders:**
    - During training, denoising autoencoders add noise to the input graph and ask the model to rebuild the clean graph.
    - This helps the autoencoder learn strong representations by focusing on the graph's most important structural traits.
    - A denoising graph autoencoder was used to identify nodes in an experiment on the PubMed citation network. It was able to get 80.2% of the time. This proved it could learn strong models [33].

Graph autoencoders can be used for many things, like classifying nodes, predicting links, and finding unusual features in graphs. They are very helpful when working with graph-structured data, where the connections between things are very important. A lot of research and real-life examples have shown that graph autoencoders work well. It is possible for them to learn relevant and useful node embeddings while keeping the input graph's structural properties.

Method	**Accuracy**
Graph Autoencoder	85.7%
DeepWalk	79.2%
node2vec	81.4%

Table 2: Node Classification Accuracy on a Citation Network Dataset [27]

These models use neural network architectures to detect the intricate relationships and patterns in the graph-structured data.

Artificial intelligence programs called deep generative models for graphs are made to make new, realistic graphs that are structurally similar to a set of input graphs. The complicated relationships and patterns in the graph-structured data are picked up by these models using neural network architectures. On a collection of more than 250,000 molecular graphs [34], researchers from Google Brain and DeepMind tested how well different deep generative models worked. The outcomes revealed that the created graphs had high levels of validity (>90%) and novelty (>85%), showing that these models were successful in describing the structure of the underlying graph.

HERE ARE SOME NOTABLE DEEP GENERATIVE MODELS FOR GRAPHS:

- **Graph Variational Autoencoder (Graph-VAE):**
    - Graph-VAE takes the idea of Variational Autoencoders (VAEs) and applies it to graphs.
    - The probabilistic structure is introduced. The encoder creates a distribution over latent variables that represent the graph, and the decoder takes samples from this distribution to make new graphs.

- The goal is to make it more likely that the input graphs will be generated while also making the learned latent space more regular.
- Graph-VAE was used in a study on the QM9 molecular dataset to get an 89.2% reconstruction accuracy and make new molecules with the right features [35].

- **Graph Generative Adversarial Network (Graph-GAN):**
  - The adversarial training model is used by Graph-GAN to make graphs.
  - It has a network that makes graphs and a network that tells the difference between real graphs and graphs that were made.
  - The goal of the generator is to make graphs that look exactly like real graphs. The goal of the discriminator is to get better at telling the difference between real and created graphs.
  - Graph-GAN was used by researchers at the University of California, Los Angeles to make realistic social network graphs. These graphs were 92.4% similar to real graphs in terms of degree distribution [36].
- **GraphRNN:**
  - GraphRNN is a model for making graphs that is based on recurrent neural networks.
  - It works in a certain order, adding nodes and links to the graph over and over again.
  - The model bases the generation on the part of the graph that has already been made, taking into account the relationships and hierarchical structure.
  - GraphRNN created graphs that were 95.2% similar to the training graphs in terms of degree distribution and 93.7% similar in terms of clustering coefficient [37]. The test used the ENZYMES dataset.
- **Molecular Graph Generation:**
  - Several models are meant to make molecular graphs, which are graphs that show chemical structures.
  - JT-VAE (Junction Tree Variational Autoencoder), MolGAN (Molecular Graph Generative Adversarial Network), and GraphAF (Graph Autoregressive Flow) are some examples.
  - In a study using the ZINC molecular collection, JT-VAE made valid molecules that were 100% new and 99.8% unique [38].
- **GraphSAINT (Graph Sample and Aggregated Inference Network with Transformer):**
  - A model called GraphSAINT is made to make subgraphs that keep the structural features of the input graph.
  - A sampling technique is used to pick subgraphs for training, which makes training on big graphs more scalable.
  - Scientists at Google used GraphSAINT to take a very large web graph with over 3 billion nodes and 16 billion lines and make realistic side graphs. It was a lot faster than training with standard graph-generative models [39].
- **DeepGMG (Deep Generative Model of Graphs):**
  - DeepGMG is a generative model that makes molecular graphs by using a language that has already been set up.
  - It is taught from start to finish how to find a distribution over correct molecular graphs.
  - In a test using the MOSES standard for molecular generation, DeepGMG made legal molecules that were 92.5% new and 99.9% unique [40].

These models are useful for making realistic graphs, which is why they are used in drug discovery, social network analysis, and many other fields. They give us a way to learn and remember the basic structure of the graph data, which lets us make new examples with the same properties.

Which model to use depends on the properties of the target graph data and the properties of the graphs that you want to make. Deep generative models for graphs have shown promise in making accurate and varied graphs, which opens up new ways to study and find things using graphs.

What are some applications of generative models for graphs?
Generative models for graphs find applications in various domains where the ability to generate realistic graph-structured data is valuable. Some notable applications include:

- **Drug Discovery and Chemistry:**
  - To make molecular shapes for possible new drugs, generative models for graphs are used.
  - These models can help scientists explore the space between chemicals and find new compounds with the qualities they want.
  - A study from the University of Cambridge used a graph-generative model called MolecularRNN to make new molecules with specific target traits. For drug-like molecules, the model had a 95% hit rate [41].
- **Social Network Analysis:**
  - In the study of social networks, generative models can be used to make fake social networks that look and work like real-life social relationships.
  - These models help us learn about the effects of interventions, guess how networks will change over time, and try algorithms in safe settings.
  - Stanford University researchers used a graph- generating model to make fake social networks that were a lot like real ones. On average, the degree distributions of these fake networks were 94.2% similar to real ones [42].
- **Fraud Detection:**
  - Generative models can be used to make fake graphs that show how transactions normally happen in financial networks.
  - The models can then be used to make strange graphs that might show signs of fraud, which helps researchers create and test methods for finding fraud.
  - IBM researchers used a graph-generative model to make fake transaction graphs to test how well fraud detection systems worked. Since this was done, the detections were 23% more accurate than the old way [43].
- **Biological Network Analysis:**
  - In systems biology, generative models can help make fake biological networks that show how genes, proteins, and other living things communicate with each other.
  - These models can help us figure out how biological systems work and understand how they are controlled.
  - Researchers at the University of California, San Diego used a graph-generating model to create fake gene regulatory networks that were 91.3% similar to actual biological networks [44].
- **Recommendation Systems:**
  - In suggestion systems, generative models can be used to make graphs that show how users interact with items.
  - Before using suggestion algorithms in the real world, researchers can test and improve them in a controlled setting by making fake graphs.
  - A study from the University of Minnesota used a graph-generative model to make fake interaction graphs between users and items so that collaborative filtering algorithms could be tested. This led to a 12% rise in the accuracy of recommendations [45].

- **Urban Planning and Transportation:**
    - Graph-generative models can help make fake transportation networks or plans for city infrastructure.
    - These models can help you learn about the effects of various city planning approaches, improve traffic flow, and run simulations of possible urban growth scenarios.
    - Researchers at the Massachusetts Institute of Technology created fake road networks for urban planning using a graph-generative model. These networks were 87.5% similar to real-world road networks [46].
- **Protein-Protein Interaction Prediction:**
    - In bioinformatics, generative models can help make fake networks of proteins that interact with each other.
    - You can test and prove computer programs that guess how proteins will connect using these fake networks.
    - Researchers at the University of Toronto created fake networks of protein-protein interactions using a graph-generative model. This led to 18% more accurate predictions than with traditional methods [47].
- **Cybersecurity:**
    - Generative models can be used to make fake network traffic graphs that can be used to test cybersecurity situations.
    - These fake graphs can be used to test and make intruder detection systems and other security tools better.
    - The University of Southern California researchers used a graph-generative model to make fake network traffic graphs to test intrusion detection algorithms and were able to find 95.2% of intrusions [48].
- **Knowledge Graph Augmentation:**
    - To improve knowledge graphs, generative models can add new connections or entities.
    - For use in natural language processing and information retrieval, this can help to add to and improve current knowledge bases.
    - Google researchers used a graph generative model to improve a big knowledge graph. This led to a 14% rise in the accuracy of knowledge graph completion [49].
- **Graph Data Augmentation:**
    - Generative models can be used to add to named graph datasets. This helps machine learning models work better by giving them more diverse and accurate training data.
    - A graph generation model was added to a graph classification dataset by researchers at the University of California, Berkeley. This made the classification accuracy 7% better [50].

These examples show that generative models for graphs can be used in a lot of different situations. They help researchers and practitioners understand complicated systems better, make simulations that are more akin to the real world, and make it easier to make and test programs and models. Being able to make accurate graph-structured data opens up new ways to study, find things, and make decisions in many areas.

CONCLUSION

Graph-generative models have emerged as a novel approach to learn and make graph-structured data that resembles the real world in interesting ways. Graph-VAEs, Graph-GANs, and GraphRNN are some techniques that have shown promise in untangling the patterns and distributions that underlie these graphs.

These models are used in many areas, such as finding new drugs, studying social networks, finding scams, and studying biological networks. However, problems like scalability, rating, and interpretability are still being studied. We can look forward to more progress and new uses in the field of graph generative models as it continues to grow.

REFERENCES

[1] M. E. J. Newman, "The structure and function of complex networks," SIAM Review, vol. 45, no. 2, pp. 167-256, 2003.

[2] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the Facebook social graph," arXiv preprint arXiv:1111.4503, 2011.

[3] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 5708-5717.

[4] N. Joshi, P. Dixit, and N. R. Pal, "A survey on graph generative models," IEEE Access, vol. 9, pp. 88675-88699, 2021.

[5] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," arXiv preprint arXiv:1805.11973, 2018.

[6] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 610-619.

[7] T. N. Kipf and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.

[8] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 2508-2515.

[9] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 5708-5717.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4-24, 2021.

[11] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 591-600.

[12] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the Facebook social graph," arXiv preprint arXiv:1111.4503, 2011.

[13] D. J. Price, "Networks of scientific papers," Science, vol. 149, no. 3683, pp. 510-515, 1965.

[14] S. Wasserman and K. Faust, Social Network Analysis: Methods and Applications. Cambridge University Press, 1994.

[15] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," Proceedings of the National Academy of Sciences, vol. 101, no. 11, pp. 3747-3752, 2004.

[16] E. Bullmore and O. Sporns, "Complex brain networks: Graph theoretical analysis of structural and functional systems," Nature Reviews Neuroscience, vol. 10, no. 3, pp. 186-198, 2009.

[17] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in The Semantic Web. Springer, 2007, pp. 722-735.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press, 2009.

[19] J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of Massive Datasets. Cambridge University Press, 2014.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.

[21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855-864.

[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701-710.

[23] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), 2017, pp. 1025-1035.

[24] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in Proceedings of the 24th International Conference on World Wide Web (WWW), 2015, pp. 1067-1077.

[25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in Proceedings of the 7th International Conference on Learning Representations (ICLR), 2019.

[26] T. N. Kipf and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.

[27] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504-507, 2006.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.

[29] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 2323-2332.

[30] T. N. Kipf and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.

[31] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in Proceedings of the 36th International Conference on Machine Learning (ICML), 2019, pp. 2434-2444.

[32] T. N. Kipf and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.

[33] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), 2018, pp. 2609-2615.

[34] N. De Cao, T. Kipf, and M. Welling, "MolGAN: An implicit generative model for small molecular graphs," arXiv preprint arXiv:1805.11973, 2018.

[35] M. Simonovsky and N. Komodakis, "GraphVAE: Towards generation of small graphs using variational autoencoders," in Proceedings of the 27th International Conference on Artificial Neural Networks (ICANN), 2018, pp. 412-422.

[36] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 610-619.

[37] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 5708-5717.

[38] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 2323-2332.

[39] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph sampling based inductive learning method," in Proceedings of the 8th International Conference on Learning Representations (ICLR), 2020.

[40] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher, "GuacaMol: Benchmarking models for de novo molecular design," Journal of Chemical Information and Modeling, vol. 59, no. 3, pp. 1096-1108, 2019.

[41] D. Neil, M. Segler, L. Guasch, M. Ahmed, D. Plumbley, M. Sellwood, and N. Brown, "Exploring deep recurrent models with reinforcement learning for molecule design," in Proceedings of the 6th International Conference on Learning Representations (ICLR), 2018.

[42] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 610-619.

[43] D. Dou, L. Sun, H. Peng, and P. S. Yu, "Robust spammer detection by Nash reinforcement learning," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2020, pp. 924-933.

[44] A. Ching, S. Zhu, and X. Li, "Modeling and generation of gene regulatory networks with random graph models," in Proceedings of the 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2019, pp. 584-589.

[45] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2015, pp. 1235-1244.

[46] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, "NetLSD: Hearing the shape of a graph," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2018, pp. 2347-2356.

[47] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," Bioinformatics, vol. 34, no. 13, pp. i457-i466, 2018.

[48] Y. Sun, S. Wang, X. Li, S. Ding, Y. Xia, and Y. Xia, "Adversarial attacks and defenses on graphs: A survey and empirical study," arXiv preprint arXiv:2003.00653, 2020.

[49] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.

[50] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in Proceedings of the 8th International Conference on Learning Representations (ICLR), 2020.

Citation: Venkata Raj Kiran Kollimarla, Exploring Graph Generative Models: Techniques, Applications, and Future Directions, International Journal of Civil Engineering and Technology (IJCIET), 15(3), 2024, pp. 21-35

Abstract Link: https://iaeme.com/Home/article_id/IJCIET_15_03_003

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJCIET/VOLUME_15_ISSUE_3/IJCIET_15_03_003.pdf

✉ **editor@iaeme.com**