STM JOURNALS

International Journal of
Algorithms Design and
Analysis Review

ISSN: 2584-1866
Volume 3, Issue 1, 2025
January–June
DOI (Journal): 10.37591/IJADAR

https://journals.stmjournals.com/ijadar

Review IJADAR

Intrinsic Evaluation of Graph Embeddings: Assessing Clustering and Community Detection Performance

Vibhu Verma[1,*]

Abstract

This paper presents an intrinsic evaluation of some graph embedding techniques on clustering and community detection tasks. We analyze a diverse set of embedding methods, ranging from traditional techniques such as Laplacian eigenmaps to more recent approaches like graph autoencoders, high-order proximity preserved embedding (HOPE), and graph attention network (GAT), using two widely studied datasets, Cora and CiteSeer. Our evaluation relies on two main metrics: Silhouette score with respect to cluster separation and Modularity regarding community detection. The results show that techniques like HOPE, GraRep, and GAT are very good at capturing community structure. Among these, GraRep seems to be the most balanced for both Silhouette and Modularity scores. Other techniques, while very good at creating well-separated clusters, such as structural deep network embedding (SDNE), perform poorly regarding community detection. In contrast, the approaches of Struc2Vec (simulated) and Poincaré exhibit poor performance along both axes. The current work has important implications for understanding strengths and limitations of different graph embedding methods for future research and applications on network analysis, social media analytics, and recommender systems.

Keywords: Graph embeddings, intrinsic evaluation, clustering, community detection, Silhouette score, Modularity, graph neural networks, HOPE, GraRep, GAT, SDNE, node2vec, DeepWalk, network analysis, community structure, machine learning, data representation

INTRODUCTION

The general trend of network embedding techniques maps each component of the graph into low-dimensional vectors while preserving certain structural and relational information of the graph. Such representations allow machine learning over graph data in an effective way [1].

Graph-based tasks are at the heart of many domains, with an ever-increasing use in drawing insights from the analysis and predictions of complex networks. Node classification can be used to predict the labels of nodes according to their features and structural properties, such as distinguishing between bots and humans in social networks. Link prediction focuses on the missing or future edges in a graph, finding new friendships among members of social networks, or predicting protein–protein interactions. Community detection refers to the formation of communities including a number of nodes based on their connectivity. It is, therefore, quite useful in applications such as finding user groups with shared interests. Graph clustering is an approach aimed at partitioning either a graph or an embedding space into clusters. Applications include segmenting citation networks into research domains. Graph

***Author for Correspondence**
Vibhu Verma
E-mail: vvibhu1@gmail.com

[1]Principal Data Scientist, Department of Computer Science, GWU, Capital One, NY, USA

Received Date: January 31, 2025
Accepted Date: February 10, 2025
Published Date: February 21, 2025

Citation: Vibhu Verma. Intrinsic Evaluation of Graph Embeddings: Assessing Clustering and Community Detection Performance. International Journal of Algorithms Design and Analysis Review. 2025; 3(1): 40–48p.

alignment concerns mapping nodes across structurally similar graphs [2]. Node similarity/ranking calculates the score or ranking of node similarities with the aim of finding influential nodes in social networks. Graph classification classifies entire graphs, for instance, categorizing molecular graphs based on their biochemical properties. Finally, anomaly detection reveals nodes, edges, or subgraphs that deviate from common patterns, which is very useful for fraud detection in financial transaction networks. These tasks are important for understanding and leveraging graph structures in a wide variety of domains, from social media analysis to biological research and beyond [3].

Evaluating graph embedding methods is essential for understanding their effectiveness in various applications [4]. The evaluation process can be broadly categorized into intrinsic evaluations, extrinsic evaluations, runtime and scalability, hyperparameter sensitivity, robustness, and task-specific metrics [5]. Intrinsic evaluations focus on properties like clustering quality and dimensionality reduction. Metrics such as Silhouette Score and Modularity measure how well embeddings separate communities. Dimensionality reduction techniques like t-SNE (t-distributed stochastic neighbor embedding) or UMAP (uniform manifold approximation and projection) visually assess how well the graph structure is preserved in lower dimensions [6].

Extrinsic evaluations test embeddings on downstream tasks like node classification, link prediction, and graph classification. These tasks help determine how useful the embeddings are for real-world applications, using metrics like accuracy, Precision@K, and AUC-ROC (area under the curve–receiver operating characteristic) [7].

Runtime and scalability evaluations compare methods based on computational efficiency, memory usage, and their ability to handle large graphs. Hyperparameter sensitivity assesses how embedding quality changes with variations in parameters like embedding size and walk lengths [8].

Robustness analysis tests how well the embeddings perform when the graph is noisy or altered. Task-specific metrics, such as F1-score for classification or NMI (normalized mutual information) for community detection, evaluate performance in specific tasks [9].

In conclusion, a comprehensive evaluation using these approaches ensures that graph embeddings are both theoretically sound and practically effective for various applications. Discussed here are two methods of intrinsic techniques in network embeddings, first being the Silhouette Score and second being Modularity [10].

Silhouette Score in Network Embeddings: Network embeddings aim to represent nodes in a graph in low-dimensional vectors while preserving some structural properties. The Silhouette Score is an intrinsic evaluation measure with respect to the quality of clustering in network embeddings [11]. Once the nodes are embedded, clustering algorithms like K-means can be used to cluster similar nodes together. The Silhouette Score provides an assessment of cohesion and separation with respect to the clusters in the embedding space:

- Cohesion is a measure of similarity for nodes within a cluster in terms of vector representation.

Separation estimates the distance between clusters. This metric ensures that nodes from different clusters are very distinct. A high Silhouette Score represents a well-separated and cohesive clustering of the network, which is an indication that the embedding algorithm captured the graph's structure pretty well. For instance, in social networks, nodes represent people [12]. Then, a high Silhouette Score within this network would imply that the embedding has managed to ascertain social patterns, keeping similar people together while keeping groups distinct [13].

Modularity in Network Embeddings: Modularity is another intrinsic evaluation measure that assesses how well the community structure of a graph is preserved in the embedding space. After applying an embedding algorithm, Modularity tells how well the embedded nodes form communities reflecting the structure of the original graph [13].

A high Modularity score means nodes within the same community are strongly connected and have fewer connections with nodes from other communities. This measure is particularly useful for tasks such as community detection. For example, in a citation network where nodes represent research papers, Modularity would say how well the clustering of papers in the embedding space reflects real research communities or topics. A high Modularity score means that the embedding has captured the community structure well [14].

EMBEDDING TECHNIQUES

Matrix Factorization

Laplacian Eigenmaps

Laplacian eigenmaps create embeddings by factorizing the graph Laplacian matrix. This method leverages the graph's structure, focusing on preserving local node proximities in a lower-dimensional space. The Laplacian matrix captures node connections, and eigenvalues and eigenvectors are computed to represent nodes. While effective for smaller graphs, this method is computationally expensive for large-scale graphs. It does not handle node attributes, and embeddings need to be recomputed for new nodes. The result is a compact representation where nearby points in the embedding space correspond to closely connected nodes in the original graph [15].

HOPE

HOPE (high-order proximity preserved embeddings) learns node embeddings by preserving higher-order proximities (e.g., Katz index or Rooted PageRank). It uses matrix factorization techniques to compute similarity matrices and decompose them into embeddings. HOPE aims to retain both the global and local structural properties of the graph, ensuring that closely related nodes remain closer in the embedding space. However, it involves computationally intensive pre-computations of similarity matrices, limiting scalability. HOPE is suited for directed and asymmetric graphs, enabling more comprehensive proximity preservation [16].

Random Walk Based

DeepWalk

DeepWalk generates random walks (sequences of nodes) from a graph to learn node embeddings. It applies the skip-gram model, commonly used in natural language processing, to these sequences, treating nodes as "words" and their connections as "contexts." This enables learning embeddings that reflect the graph's local and global structure. DeepWalk is computationally efficient, scalable, and suitable for large graphs. However, it ignores node attributes and requires retraining for new nodes. The learned embeddings capture community structures and relationships effectively, making it a popular choice for link prediction and clustering tasks [17].

Node2vec

node2vec extends DeepWalk by introducing biased random walks, balancing local (breadth-first search) and global (depth-first search) exploration. This flexibility enables it to capture both community structure and structural equivalence in graphs. It uses hyperparameters p and q to control the likelihood of revisiting nodes or exploring farther ones. The sequences generated by biased random walks are input to the skip-gram model for learning embeddings. node2vec improves over DeepWalk by allowing more nuanced graph exploration, but its effectiveness depends on careful hyperparameter tuning.

Struc2vec

Struc2vec learns embeddings focused on structural equivalence, where nodes with similar roles in the graph (e.g., hubs) are mapped closer together, regardless of direct connections. It first builds a multi-layer weighted graph representing pairwise structural similarities, then performs hierarchical random walks over this structure. The resulting node sequences are used for training embeddings via skip-gram. Struc2vec differs from DeepWalk and node2vec by prioritizing structural roles over proximity. While effective for tasks like graph comparison, its computational complexity makes it less suitable for very large graphs.

Graph Convolutional Neural Networks

Graph Convolutional Neural Networks (GCNs)

GCNs generalize convolutional neural networks to graph data. They aggregate features from a node's neighbors using graph convolutions, combining the node's own attributes and its neighbors' features. This iterative aggregation enables GCNs to capture both local and global patterns in graphs. GCNs require labeled data for supervised learning and are effective for semi-supervised tasks, like node classification. However, they struggle with scalability in dense or large graphs due to computational demands. GCNs excel in handling attributed graphs and incorporating node and edge features into embeddings.

GraphSAGE

GraphSAGE (Sample and Aggregate) improves scalability by sampling a fixed-size neighborhood for each node and aggregating their features. Unlike GCNs, GraphSAGE learns to generate embeddings inductively, making it suitable for new, unseen nodes during inference. The aggregation function can be mean, long short-term memory (LSTM)-based, or pooling, allowing flexibility in capturing graph structures. By avoiding full-graph convolutions, GraphSAGE is more efficient for large, dynamic graphs. However, it may lose some detail in densely connected graphs due to its sampling mechanism.

Graph Attention Networks (GATs)

GATs use attention mechanisms to dynamically weigh the importance of neighboring nodes during feature aggregation. Each edge is assigned an attention score, enabling the network to prioritize informative neighbors while down-weighting less relevant ones. This adaptivity makes GATs versatile for graphs with varying densities and complex structures. They are inductive and capable of handling attributed graphs. However, GATs are computationally intensive, especially for large graphs, due to the overhead of attention score computations for each edge.

Deep Graph Infomax (DGI)

DGI is an unsupervised method that learns node embeddings by maximizing mutual information between local (node-level) and global (graph-level) representations. It creates a corrupted version of the graph as a negative sample and trains the model to distinguish the true graph from the corrupted one. This helps DGI capture meaningful features without requiring labels, making it ideal for unsupervised learning tasks. While versatile and effective for smaller graphs, DGI may face challenges with scalability.

Edge Prediction-Based

Structural Deep Network Embedding (SDNE)

SDNE uses a deep autoencoder to learn node embeddings that preserve both first-order (local) and second-order (global) proximities. The encoder maps nodes to embeddings, and the decoder reconstructs the graph's adjacency matrix. A combination of reconstruction loss and Laplacian regularization ensures embeddings capture the graph's structure effectively. SDNE is particularly suited for graphs with complex structures but is computationally demanding and requires retraining for new nodes.

GraRep

GraRep captures node proximities across multiple scales (e.g., 1-hop, 2-hop) using matrix factorization. It decomposes different k-step proximity matrices to compute embeddings that integrate information from various scales. GraRep is effective for static graphs but lacks support for node attributes and struggles with scalability due to its reliance on matrix factorization.

Matrix Completion

Graph Auto Encoders (GAEs)

GAEs employ autoencoders to learn embeddings by reconstructing the adjacency matrix. The encoder maps nodes to a latent space, while the decoder reconstructs node connections. GAEs work well for simple graph structures but have limited scalability. They often require retraining for updates, making them less suitable for dynamic graphs (Table 1).

Table 1. Summary of clustering techniques.

Clustering Technique	Pros	Cons
Spectral clustering	Captures non-linear relationships, good for graph-structured data.	Expensive for large graphs; scales poorly without preprocessing.
K-means (with principal component analysis [PCA])	Scalable, efficient, and works well after dimensionality reduction.	Assumes spherical clusters, sensitive to initialization, may not capture complex structures.

Graph Reconstruction

Variational Graph Autoencoders (VGAE)

VGAEs extend GAEs by incorporating variational inference, enabling probabilistic embeddings that capture uncertainty in graph structures. They reconstruct graphs by sampling from a latent distribution, making them flexible for complex graph structures. However, VGAEs are computationally expensive and require retraining for changes in the graph.

Poincaré Embeddings

Poincaré embeddings map nodes to hyperbolic space, which is well-suited for capturing hierarchical structures in graphs. By leveraging the hyperbolic geometry, it represents tree-like graphs efficiently. While effective for such graphs, the mathematical complexity and hyperbolic computations make it less practical for general-purpose use.

EXPERIMENTAL DESIGN

This work investigates the performance of various network embedding methods such as Laplacian Eigenmaps, HOPE, DeepWalk, node2vec, Struc2Vec (Simulated), GCN, GraphSAGE, GAT, DGI, SDNE, GraRep, Graph Autoencoder, Graph Autoencoder Decoder, and Poincaré on two benchmark citation datasets – Cora and CiteSeer. Intrinsic evaluations are done based on Silhouette Score and Modularity to find the quality of clustering and community preservations in embeddings. Silhouette Score will give us an idea of the cohesion of nodes within clusters, while Modularity will evaluate how well the community structure of the original graph is preserved in the embedding space. Node embedding techniques are tested by generating node embeddings, running clustering algorithms (K-means or Spectral Clustering), and comparing the results on both datasets. HOPE, Graph SAGE, GCN, and GAT are expected to work well in preserving community structure and generating well-separated clusters, while methods like Struc2Vec (Simulated) and Poincaré may underperform. This experiment will give a good insight into the effectiveness of these techniques for tasks like community detection and node classification, hence finding the most suitable methods for graph-based applications.

Evaluation metrics such as Silhouette Score and Modularity are essential for assessing the quality of network embeddings. The Silhouette Score measures how similar a data point is to its own cluster compared to others, operating directly in the embedding space, whether high-dimensional or reduced. It ranges from –1 to 1, where a score of 1 indicates perfectly separated clusters, 0 indicates overlapping clusters, and negative values suggest misclassified points. On the other hand, Modularity evaluates the strength of a graph's division into communities, requiring the mapping of clustering results back to the graph nodes. A high modularity score (greater than 0.3) signifies a well-defined community structure, indicating that the clustering preserves the graph's inherent connectivity patterns. Both metrics are crucial for evaluating the effectiveness of embedding techniques in capturing the graph's structural and community properties (Tables 2 and 3).

RESULTS

- *Method Type:* The broad category of the approach, grouping methods based on their fundamental technique (e.g., matrix factorization, random walk, etc.).
- *Method Name:* The specific name of the algorithm or method (e.g., DeepWalk, GCN, etc.).
- *Short Explanation:* A concise description of how the method works and its purpose.
- *Uses Graph Structure Only / Graph + Attributes:* Specifies whether the method uses only the graph's structure (edges) or includes node/edge attributes (features).

- *Pros and Cons:* Highlights the advantages and limitations of the method for better evaluation.
- *Can Generate Embeddings on New Data Without Rerunning:* Indicates if the method can generate embeddings for unseen nodes/edges without retraining.
- *Incremental Learning:* Refers to the method's ability to update its model with new data without retraining from scratch.
- *Node Embedding Fine-Tuning:* Explains if the method allows further adjustment of embeddings for specific tasks.
- *Inductive Learning:* Describes whether the method can generalize to unseen graph data during inference.
- *Precompute Static Embeddings:* States if the method requires precomputing embeddings for the graph before applying to tasks.
- *Parallelization:* Indicates if the method supports parallel computation for faster processing.
- *Graphics Processing Unit (GPU) Acceleration:* Specifies whether the method can leverage GPUs for performance optimization.

Table 2. Summary of Cora results.

Embedding Technique	Silhouette K-Means	Silhouette Spectral	Modularity K-Means	Modularity Spectral
Laplacian eigenmaps	0.0562	0.0111	0.1080	0.0539
HOPE	0.0714	-0.1120	0.5807	0.1871
DeepWalk	0.0869	-0.0502	0.0153	0.0016
node2vec	0.0873	-0.0024	0.0078	0.0050
Struc2Vec (Simulated)	0.0110	0.0021	0.0028	-0.0044
GCN	0.0471	0.0041	0.5309	0.4432
GraphSAGE	0.0230	-0.0004	0.1340	0.0198
GAT	0.0325	-0.0252	0.5815	0.0826
DGI	0.0248	0.0074	0.2274	0.3176
SDNE	**0.1353**	0.0386	0.0438	0.0104
GraRep	0.1098	-0.0824	0.2882	0.1682
Graph Autoencoder	0.0724	-0.0223	0.3806	0.3695
Graph Autoencoder Decoder	0.0880	0.0390	0.3818	0.2841
Poincaré	0.0116	0.0019	0.0007	-0.0088

Table 3. Summary of CiteSeer results.

Embedding Technique	Silhouette K-Means	Silhouette Spectral	Modularity K-Means	Modularity Spectral
Laplacian Eigenmaps	0.0628	0.0039	0.0870	0.0030
HOPE	0.2559	0.1676	0.1284	0.0350
DeepWalk	0.0694	-0.0534	0.0009	-0.0006
node2vec	0.0762	-0.0270	0.0090	-0.0079
Struc2Vec (Simulated)	0.0112	0.0019	-0.0007	-0.0058
GCN	0.0352	-0.0454	0.5217	0.0865
GraphSAGE	0.0211	-0.0401	0.1790	0.1842
GAT	0.0175	-0.0125	0.6107	0.4255
DGI	0.0238	-0.0407	0.3214	0.0623
SDNE	0.1533	0.1840	0.0550	0.0409
GraRep	0.2759	0.1913	0.1420	0.4765
Graph Autoencoder	0.1086	-0.1015	0.3499	0.4196
Graph Autoencoder Decoder	0.1046	0.0683	0.5822	0.3368
Poincaré	0.0108	0.0015	-0.0019	-0.0088

Table 4. Summary of techniques and important things to consider.

Method Type	Method Name	Uses Graph Structure Only / Graph + Attributes	Can Generate Embeddings on New Data Without Rerunning	Incremental Learning	Node Embedding Fine-Tuning	Inductive Learning	Precompute Static Embeddings	Parallelization	GPU Acceleration
Matrix Factorization	Laplacian Eigenmaps	Graph Structure Only	No (requires retraining)	No	No	No	Yes	No	No
	HOPE	Graph Structure Only	No (requires retraining)	No	No	No	Yes	No	No
Random Walk-Based	DeepWalk	Graph Structure Only	No (requires retraining)	No	No	No	Yes	Yes	Yes
	node2vec	Graph Structure Only	No (requires retraining)	No	No	No	Yes	Yes	Yes
	Struc2vec	Graph Structure Only	No (requires retraining)	No	No	No	Yes	Yes	Yes
Graph Neural Networks	GCN	Graph + Attributes	Yes (inductive)	Yes	Yes	Yes	No	Yes	Yes
	GraphSAGE	Graph + Attributes	Yes (inductive)	Yes	Yes	Yes	No	Yes	Yes
	GAT	Graph + Attributes	Yes (inductive)	Yes	Yes	Yes	No	Yes	Yes
	DGI	Graph + Attributes	Yes (inductive)	Yes	Yes	Yes	No	Yes	Yes
Edge Prediction-Based	SDNE	Graph Structure Only	No (requires retraining)	No	No	No	Yes	Yes	Yes
	GraRep	Graph Structure Only	No (requires retraining)	No	No	No	Yes	No	No
Matrix Completion	Graph Autoencoders (GAE)	Graph Structure Only	No (requires retraining)	No	No	No	No	Yes	Yes
Graph Reconstruction	Variational Graph Autoencoders	Graph + Attributes	No (requires retraining)	No	No	No	No	Yes	Yes
	Poincaré Embeddings	Graph Structure Only	No (requires retraining)	No	No	No	No	Yes	Yes

Among the different graph embedding techniques compared on the Cora and CiteSeer datasets, GraRep, HOPE, and GAT are the best techniques with the highest Silhouette Score and Modularity (Table 4). The most balanced technique is GraRep, with a good value for both metrics. It has a very strong Silhouette Score (0.2759 for K-means) and Modularity (0.4765 for Spectral), indicating not only that the clusters are well separated but it also captures the community structure within the data well. Therefore, GraRep is the best overall technique.

While HOPE represents an extreme case in Modularity, especially regarding K-means clustering (0.5807 for Cora, 0.1284 for CiteSeer), its Silhouette Score is very low, especially for the case of Spectral clustering. Thus, HOPE seems to identify the clusters rather effectively but not very sharp, and also does not manage to preserve the global graph structure.

GAT also performs very well on Modularity, 0.6107 for CiteSeer, which proves its ability in detecting strong communities. However, the Silhouette Score is low, which indicates weak cluster separation. That

means while GAT is effective in capturing the local communities, it is not effective in clearly separating them into distinct clusters.

GCN and Graph Autoencoder Decoder both did relatively well in Modularity. GCN captured both local and global structures of the graph, even though its Silhouette Scores were just moderate. Graph Autoencoder Decoder has the best Modularity score in CiteSeer, 0.5822; hence, this model should serve well for community detection, although it is not outstanding in clustering.

The methods that fared better were SDNE, DeepWalk, and node2vec, which showed relatively more moderate Silhouette Scores. Although their Modularity results are not quite impressive, they generate reasonable clustering results but are less effective in the capturing of community structure. The poorest performing methods for both metrics have been Struc2Vec (Simulated) and Poincaré. It suggests that these methods poorly preserve meaningful structure within the graph.

Following are the key takeaways:

a. *Best overall performer:* GraRep (balanced performance in both clustering and community structure).
b. *Best for Modularity:* HOPE and GAT (strong community detection but weak cluster separation).
c. *Best for clustering:* SDNE (good separation but poor in community detection).
d. *Weakest performers:* Poincaré and Struc2Vec (Simulated) (struggling in both clustering and community structure).

CONCLUSION

GraRep is the most versatile and effective technique for graph embedding across both datasets, excelling in both Silhouette Score and Modularity. HOPE and GAT shine in community detection but require further refinement for clearer cluster separation. SDNE, while strong in clustering, does not perform well in community detection, and methods like DeepWalk and node2vec are better suited for specific clustering tasks but lack robustness in community detection. Struc2Vec (Simulated) and Poincaré have limited effectiveness and should be considered less favorable for graph embedding tasks.

REFERENCES

1. Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Dietterich TG, Becker S, Ghahramani Z, editors. NIPS 2001 – Advances in Neural Information Processing Systems 14, Cambridge, MA, USA: MIT Press; 2001.
2. Mohan A, Venkatesan R, Pramod KV. A scalable method for link prediction in large real world networks. J Parallel Distrib Comput. 2017; 109: 89–101.
3. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, August 24–27, 2014. pp. 701–710.
4. Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13–17, 2016. pp. 855–864.
5. Ribeiro LF, Saverese PH, Figueiredo DR. struc2vec: Learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, Nova Scotia, Canada, August 13–17, 2017. pp. 385–394.
6. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint. arXiv:1609.02907. September 9, 2016.
7. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: NIPS 2017 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, December 4–9, 2017. pp. 1025–1035.
8. Liu Z, Zhou J. Graph attention networks. In: Introduction to Graph Neural Networks. Cham, Switzerland: Springer International Publishing; 2020. pp. 39–41.

9. Veličković P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD. Deep graph infomax. arXiv preprint arXiv:1809.10341. September 27, 2018.
10. Zhu W, Wang X, Cui P. Deep learning for learning graph representations. In: Pedrycz W, Chen S-M, editors. Deep Learning: Concepts and Architectures. New York, NY, USA: Springer; 2020. pp. 169–210.
11. Cao S, Lu W, Xu Q. Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, Melbourne, Victoria, Australia, October 18–23, 2015. pp. 891–900.
12. Kipf TN, Welling M. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308. November 21, 2016.
13. Nickel M, Kiela D. Poincaré embeddings for learning hierarchical representations. In: NIPS 2017 – 31st Conference on Advances in Neural Information Processing Systems, Long Beach, CA, USA, December 4–9, 2017. pp. 6341–6350.
14. Kota TK, Rongala S. Implementing AI-driven secure cloud data pipelines in Azure with Databricks. Nanotechnol Percept. 2024; 20 (S15): 3063–3075.
15. Rongala S. An analytical review of not only SQL (NoSQL) databases: importance and evaluation. Int J Commun Netw Inform Security. 2023; 15 (3): 378–379.
16. Rongala S. An overview of key principles of effective data visualization. Int J Intell Syst Appl Eng. 2024; 12 (4): 4847–4853.
17. Joshi N. Optimizing real-time ETL pipelines using machine learning techniques. 2024. DOI: http://dx.doi.org/10.2139/ssrn.5054767

