Engineered Science
DOI: https://dx.doi.org/10.30919/es1406

Multi-Head Variational Graph Autoencoder Framework for Link Prediction on Citation Graphs

Anindyadeep Sannigrahi,[1] Rahee Walambe[2, 3,*] and Ketan Kotecha[2, 3,*]

Abstract

Graph neural networks (GNNs) are powerful tools for link prediction in diverse applications such as recommendation systems, drug discovery, knowledge graph completion, and more. Link prediction estimates the likelihood of node connections, enabling various downstream tasks and providing insights into network structures. However, generating highly representative node embeddings in a semi-supervised setting with limited labelled data is challenging. GNNs, utilizing neural message passing, effectively leverage local and global graph information to address this issue. Our research aims to enhance node embeddings for link prediction by developing novel techniques that capture intricate graph patterns, incorporating both topological and semantic information. We leverage the expressive power of GNNs to overcome the limitations of traditional methods and improve link prediction accuracy. The proposed model was tested for three benchmark datasets, namely Cora, CiteSeer, and PubMed, for the link prediction tasks on citation graphs. It is observed that the proposed model outperforms the existing models for this task by achieving the area under curve (AUC) and average precision (AP) scores of 99.1%, 99.2% for Cora, 98.5%, 98.3% for Citeseer, and 99.1%, 98.9% for the PubMed dataset, respectively. The average improvement of 2% and 3% is observed in AUC and AP scores, respectively. The proposed model outperforms the existing methods and can be extended further for other datasets.

Keywords: Graph autoencoders; Graph machine learning; Variational graph autoencoders; Graph embedding; Multi-head framework.
Received: 29 March 2024; Revised: 16 August 2024; Accepted: 27 August 2024.
Article type: Research article.

1. Introduction

Recommendation systems are important in several business aspects. Link prediction is the basis of the recommendation system when it comes to framing the recommendation problem into a graph paradigm. Traditional recommendation systems are generally based on memory-based collaborative filtering approaches or model-based collaborative filtering, like matrix factorization. However, those methods are inefficient when it comes to handling big data. At the same time, those methods are not able to capture a large context of connectivity between the items/users. Graph neural networks (GNNs) provide a solution to these issues since they can capture the non-Euclidean complexity and can capture the context of the links between the items by representing them as graphs. GNNs have gained huge popularity in recent years, primarily due to their capacity to solve many real-world problems based on non-Euclidean paradigms,[1] with some applications such as traffic prediction, drug discovery, neural machine translation, *etc*.

GNNs are very efficient in generating accurate and powerful embeddings.[2] The embeddings generated by these nodes of the GNNs form the basis of several graph-related supervised, unsupervised, and semi-supervised tasks such as node classification,[3] graph classification,[4] and link prediction,[5] *etc*. One such important task is link prediction.[6] In this work, this problem is defined and framed as a binary classification problem, where the network predicts the potential probability of the existence of a link between two graph networks. To predict the potential link between various nodes of the graph, embeddings are to be generated. There are different ways to generate embeddings of the nodes of the graph. These methods include traditional approaches like deep

[1] *School of Computer Science and Engineering, Kalinga Institute of Industrial Technology, Bhubaneshwar, Odisha, 751024, India*
[2] *Symbiosis Institute of Technology (SIT), Symbiosis International (Deemed University), Pune, 412115, India*
[3] *Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis International (Deemed University), Pune, Maharashtra, 412115, India*
**Email:* rahee.walambe@sitpune.edu.in (R. Walambe), director@sitpune.edu.in (K. Kotecha)

walk,[7] which is based on random walks, Node2Vec,[8] and MetaPath2Vec,[9] *etc*. These are transductive approaches to learning the embeddings of the node, which means every time a new node is introduced, it has to be re-trained to perform the required tasks. The random walk-based approaches are inefficient when solving real-world graph-related problems. Another class of algorithms that are efficient in learning embeddings of the nodes of the graph is based on the message-passing-based approach. Some of the popular models that follow this type of learning approach are GraphSage,[10] graph convolutional networks (GCNs),[11] graph attention networks (GATs),[12] *etc*. These methods are found to be excellent in effectively capturing the local and global information of the graph data.[13] Thus, using these approaches specifically to link prediction types of tasks produces great results. One of the very effective architectures employed for this is the variational graph autoencoder (VGAE).[14] This method is simple and efficient at the same time to perform link prediction through graph reconstruction. This autoencoder consists of two parts, an encoder and a decoder. The encoder part consists of a GCN layer followed by two more GCN layers to create μ and log(σ) for sampling the latent features. These sampled features undergo the reparameterization process to form the latent vector. This latent vector is then passed through the decoder. The decoder is simple, as it is the inner dot product between the latent vector and its transpose. This generates an N × N matrix (where N is the total number of nodes) where the higher value (close to 1) of an element (*i, j*) depicts that there exists a potential connection between node *i* and node *j*. The lower value depicts that there is no such connection between the nodes. Several other methods based on the variants of the autoencoder architecture are introduced, where the major modifications are proposed in the encoder part.[10,12] It has been observed that different types of layers, such as GCN and GAT, are used to learn the embeddings of the nodes.[15]

Many real-world tasks are achieved by machine learning (ML). A vast variety of ML algorithms and methods are proposed and implemented for various tasks. One of the important tasks is the recommendation system based on link prediction. One very interesting use case of link prediction tasks is creating recommendations, analyzing social networks, analyzing scene graphs, etc. Several approaches to link prediction-based recommendation systems are proposed. However, ML-based paradigms such as collaborative filtering are traditionally reported for this task. Researchers have also explored graph ML approaches for achieving this task in recent years. Classical methods typically include model-based collaborative filtering methods like matrix factorizations.[16] The matrix factorization algorithm is primarily based on classic dimensionality reduction.[16] This method has been really popular, and several of its variants are used to design many earlier recommendation-based problems. For example, Ma *et al*.[17] created the user and item latent feature space using probabilistic matrix factorization-based methods for social recommendations. Particularly in the recommendation systems paradigm, confidence-aware matrix factorization systems are proposed and could predict potential recommendations based on the model's confidence.[18] Data sparsity, especially on the datasets for the recommendation tasks, is a major problem. In another case,[19] authors introduced the dual-regularized matrix factorization method using deep neural networks (DRMF) to deal with this problem. In DRMF, a convolutional neural network and a gated recurrent neural network are stacked together to represent independent embeddings of users and items for recommendation predictions.[20,21] In another case,[22] fast nonparametric matrix factorization methods for large-scale recommendation systems on big data are proposed. This memory-based collaborative filtering method does not use any learnable parameters for the prediction. The probabilistic approach was also proposed,[23] also known as the probabilistic robust matrix factorization (PRMF) method, which tries to make the matrix factorization method more robust and outperforms in terms of accuracy and other evaluation metrics in the areas of both synthetic data and computer vision-based tasks. The primary disadvantage of these methods is their inability to deal with data belonging to non-Euclidean space.

When data consists of images and texts, there exists a certain Euclidean relationship, but when it comes to establishing relations within different entities, evaluating those entities on the grounds of Euclidean space is certainly not enough and is inefficient at the same time. The graph-based approaches are powerful in handling non-Euclidean data. There are different types of network/graph data on which link prediction-based tasks can be performed. Some include knowledge graphs, social graphs, scene graphs, user-item-based bi-partite graphs, citation graphs, *etc.*, specific for recommendation-based purposes. At the same time, there are two different classes of approaches for computing link predictions using graphs. One is the non-GNN-based approach, and the other is based on some variants of GNN at its core. Within the classical methods for link prediction-based tasks, multiple approaches are proposed in the literature. For example, a biased random walk with a restart is proposed.[24] The main aspect of this work is to share biased probabilities to perform the random walks to explore deeper into the underlying topological properties of the given graphical data and outperform some of the baseline results. The method of joint link prediction and network alignment via cross-graph embedding is proposed. It demonstrates the mutual benefits of link prediction and network alignment, improving the recall for both tasks.[25] A modified version of the Deepwalk algorithm to cover a greater context of the graph's nodes and topology is proposed.[7,26] Using the same methods, they proved that it is better than the other baselines. An approach for link prediction that captures the potential adversarial attacks faced by the models and discusses the privacy issues on the grounds of these adversarial attacks is proposed.[27] However, non-GNN-based graph methods are transductive and perform well on pre-existing nodes but struggle with new,

unseen nodes. Inductive-based methods employing GNNs were introduced for link prediction tasks to address this.

Recent research focuses on both static and dynamic link predictions, with robust and efficient models proposed in these areas. For example, multi-level GCNs are proposed for link predictions for cross-platform account matchings.[24] Multi-level graph convolutional networks are used to solve the problems with memory for large and sparse connections and achieve remarkable results in link predictions.[28] GCNs are also used in biological networks for link predictions to achieve increasing attention in link prediction for biology-based graph structural data.[29] Long short-term memory networks (LSTMs) are combined and very much popular in handling temporal-based data,[30] like speech recognition,[31] with GNNs and the hybrid model being used to come up with robust link prediction tasks performed on Dynamic graphs. Also, link prediction on weighted dynamic graphs is carried out using the combination of GCNs and generative adversarial networks (GANs).[32,33] Variational graph autoencoders or VGAEs are a remarkable development in the link prediction task, especially for citation-based graphs. It achieves outstanding scores of mean average precision of 92.6, 92.0, and 94.7 in citation graph datasets,[14] like Cora, Citeseer, and Pubmed,[34] respectively. Several modifications to this architecture are proposed to fine-tune the results even more. For example, Tran *et al*.[35] proposed the multi-task graph autoencoder, which not only associates with link prediction but also solves the problem of node classification simultaneously. Specifically, the penultimate decoder layer is used for semi-supervised node classification, whereas the last layer is used to reconstruct the neighborhood for supervised link prediction. The method of Graph InfoClust is proposed.[36] It mainly focuses on learning the representation from cluster-level information content using a differentiable K-means method and optimizing the results by maximizing information between nodes of the same clusters. This model is one of the state-of-the-art models for the Citeseer dataset.[34] One of the very efficient models of recent days, specific to citation network graphs, is the model named variational graph normalized auto-encoders (VGNAEs) by Seong Ahn and Kim.[37] They mainly focus on link predictions for nodes whose degrees are zero, *i.e*., for isolated nodes. This paper uses L2 normalization to derive a better representation for isolated nodes to improve the overall results. VGNAEs outperform several of the pre-existing architectures based on VGAE.[14] Specific to the link prediction task for the Citation network, the present state-of-the-art model is WalkPooling by Pan *et al*.[38] This model employs random walks and tries to combine the topological heuristics with the feature learning ability of the model, and when combined and trained with GNNs, it outperforms all the latest models out there and is the current state of the art for the Cora and Pubmed datasets, respectively.[34]

In this work, we propose a method that utilizes both the power of GCNs and GATs by combining the learned embeddings produced by GCNs and GATs independently. These combined embeddings are passed via sampling and reparameterization to obtain the resultant embedding. Those reparameterized embeddings are then passed to the inner dot product decoder to perform further link prediction tasks. The main motivation for such an approach is to combine the power of both spectral and attention features.[15] The combined architecture utilized the learned features from both GCN and GAT so that the power of these features can be leveraged by optimizing their performance with respect to the loss obtained from the combination of the contributions of both methods. In summary, the contributions of this work are:

- Developed and demonstrated the method to combine the individual embeddings of both GCNs and GATs for more powerful representations;
- Proposed a method to combine the spectral and attention features for more robust and improved link prediction;
- Evaluated and compared the proposed approach with the pre-existing methods for three standard benchmark datasets.

2. Methods

2.1 Datasets

Three benchmark citation datasets, namely Cora, Citeseer, and Pubmed are employed for the demonstration of the proposed method.[34] Cora is a dataset in the fields of computer science and information retrieval. It consists of research papers classified into seven classes based on the subject matter. Citeseer, similar to Cora, contains research papers widely used for citation analysis and clustering. Pubmed, on the other hand, is a large biomedical literature database that contains over 30 million citations and datasets.[34] However, we have taken a small subset of the Pubmed dataset for testing. The compact information of these three datasets is shown in Table 1.

Table 1: Dataset information used for training and evaluation.

Dataset	Type	Nodes	Edges	Features
Cora	Citation	2,708	5,429	1433
Citeseer	Citation	3,227	4,723	3703
Pubmed	Citation	19,717	44,338	500

Although for this work, these three datasets are considered, the model proposed is generalized and robust to any other methods. It can be scaled to larger datasets, provided the required processing and compute power are available.

2.2 Proposed model

In this paper, we propose a multi-head variational graph neural network (MVGAE). MVGAE has been a derivative of VGAE.[14] Hence, similar to VGAEs, the proposed model has two parts: an encoder and a decoder.

The encoder of this variational graph autoencoder uses a multi-headed approach, where each head is a GNN layer, from Fig. 1, it can be seen that the input $\boldsymbol{X}$ and adjacency matrix $\boldsymbol{A}$ are passed through multiple heads, in which some heads are made of GCN layers and other heads are made of multi-headed

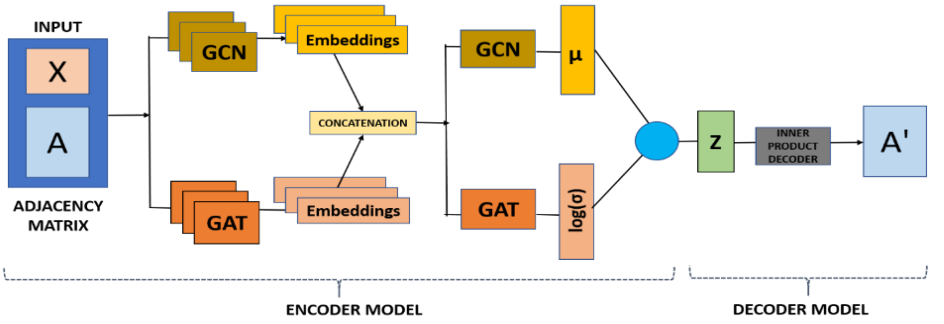

Fig. 1: Proposed framework with multi-head variational graph autoencoder, where $\boldsymbol{X}$ is the feature matrix of the nodes of the graph, and $\boldsymbol{A}$ is the adjacency matrix of the graph.

GAT layers. One GCN layer and one GAT layer are employed in our original implementation. The GAT layer has 2 to 4 attention heads. Further, the hidden embeddings that are generated from those multi-headed layers get concatenated into a single embedding vector. The concatenated vector is again passed on to one GCN block and one GAT block to generate the reparametrized variables μ and log(σ), respectively.[11,12] The latent representation $\boldsymbol{Z}$ is obtained from the reparameterization method.[39,40]

The decoder network is a simple sigmoid-based inner dot product, which is the same as VGAEs.[14] This multi-head-like fashion is inspired by Transformers and ResNet.[41] Mathematically, a graph G with a node feature matrix $X \in \boldsymbol{R}^{N \times F}$ and adjacency matrix $A \in \boldsymbol{R}^{N \times N}$, which is being normalized. The latent representation from the encoder is $\boldsymbol{Z} \in \boldsymbol{R}^{N \times (n_1 + n_2)F'}$. The inner dot product decoder takes this vector $\boldsymbol{Z}$ and carries out the inner dot product with the transpose of itself. Therefore, it becomes $(\boldsymbol{Z}.\boldsymbol{Z}^{\boldsymbol{T}}) \in \boldsymbol{R}^{N \times N}$. Further, it is passed to a sigmoid activation function. The output of the decoder is the reconstructed adjacency matrix. The (*i, j*)th element of the reconstructed adjacency matrix represents the probability of a potential link between node *i* and node *j*.

2.3 Mathematical formulation of the proposed architecture in a generalized form

Instead of picking just one GCN layer, multiple and different GNN-based layers, namely, GCN and GAT, can be extended simultaneously in parallel.[11,12] In this section, a generalized mathematical premise is provided for the same. Each of the hidden feature vectors generated by the GCN and GAT layers would belong to a vector H ∈R^(N × F'). These vectors are concatenated such that the resultant hidden matrix becomes: H_concat ∈ R^(N × (n_1 + n_2)F'), where n_1 is the number of GCN layers and n_2 is the number of GAT layers. This concatenated matrix is then passed through two more GNN layers (one GCN and the second GAT layer) to obtain the generated μ and log(σ), respectively.[42] Then the reparameterization is applied such that the final embedding Z ∈ R^(N ×(n_1 + n_2) F') is computed. The 2-layer encoder will consist of the first layer with several parallel branches called the heads. The next layers have two GNNs to compute the generated μ and log(σ), respectively.

Passing the input $\boldsymbol{X}$ to this series of parallel hidden layer architectures will generate a set of hidden embeddings coming out from GCNs and a set of hidden embeddings coming out from GATs. Those sets can be defined as in Eqs. (1-3):

$$H_{GCN} = \{ H^1_{GCN}, H^2_{GCN}, \ldots, H^{n_1}_{GCN} \} \tag{1}$$

$$H_{GAT} = \{ H^1_{GAT}, H^2_{GAT}, \ldots, H^{n_2}_{GAT} \} \tag{2}$$

$$H^1 = \left(||_{K=1}^{n_1} H^k_{GCN} \right) || \left(||_{K=1}^{n_2} H^k_{GAT} \right) \tag{3}$$

where H_{GCN} and H_{GAT} represent hidden GCN and GAT layers, respectively. H^1 from Eq. (3) is being passed through two more layers for the sampling and reparameterization, one of which is a GCN layer, and the other is a GAT layer.[11,12] Hence, the second step in the encoding process is defined by Eqs. (4) and (5):

$$\mu = \mathrm{GCN}_{\mu}(\mathrm{H}^1) \tag{4}$$

$$log(\sigma) = GAT_{log\sigma}(H^1) \tag{5}$$

where both $\{\mu, \sigma\} \in \boldsymbol{R}^{N \times (n_1 + n_2)F'}$, F' is some number of dimensions assigned. Subsequently, the latent representation is defined by Eq. (6):

$$Z = \mu + \epsilon\, \sigma \; where \; \epsilon \sim N\,(0, 1) \tag{6}$$

GCN fundamentally works on the principles of spectral methods.[11] The GCN convolution can be expressed by Eqs. (7) and (8):

$$g_\theta . x \approx \sum_{k=0}^{K} \theta'_k T_k(\check{L}) x \tag{7}$$

$$\check{L} = \frac{2}{\lambda_{max}} L - I_N \tag{8}$$

where g_θ is a kernel (θ represents a learnable parameter), and this is applied to a graph signal x. K represents the Kth-order neighborhood. T denotes the Chebyshev polynomials applied on $\check{L}$, which is the normalized graph Laplacian matrix. λ_{max} denotes the largest eigenvalue of L. Finally, the aggregated features Z are expressed by Eq. (9):[43]

$$Z = \tilde{D}^{\frac{-1}{2}} \tilde{A} \tilde{D}^{\frac{-1}{2}} X \Theta \tag{9}$$

where $\tilde{A} = A + I$, $\tilde{D}$ is the diagonal matrix. The $\tilde{D}^{\frac{-1}{2}}$ is responsible for the renormalization.[11] Hence, GCNs learns the features from the spectral graph paradigm. However, the

working of GCNs can also be considered to fit within the spatial domain based on its mechanism for message passing. On the other hand, GATs are fundamentally spatial in nature (Eqs. (7)-(9)).[12] GATs are popular for their robustness and simplicity. In conclusion, GATs can learn the underlying graph representation from the spatial paradigm. In the proposed approach in this work, consisting of Multi-Head VGAEs, the hidden representations are captured using both the spectral and spatial paradigms, and trying to leverage the characteristics of both GCNs and GATs.[11,12] The learned hidden representations of both GCNs and GATs in Eq. (6) are concatenated. Combining the features learned independently from both spectral and spatial domains helps not only to achieve better metrics but also to accelerate the training process. The significant acceleration of the training process could be achieved due to the presence of heads, where each head is either a GCN or a GAT layer, and they can compute the embeddings independently.[44] This independent computation harnesses the power of graphics processing units (GPUs) and parallelization, hence speeding up the process. This completes the encoding process to generate the required latent representations.

3. Results and discussion

3.1 Experiment design and target outcome

To compare and demonstrate the effectiveness of the proposed framework, four sets of experiments and comparisons were carried out. Two different model configurations (Configurations 1 and 2) for the training and evaluation of the model are employed. The stepwise procedure for carrying out experiments is shown in Fig. 2. The standard process includes extracting the dataset and then carrying out data preprocessing, where the masking of nodes and edges is carried out. This is followed by extracting the feature matrix of nodes and the adjacency matrix. Then, the proposed MVGAE model is trained, and the probability of a potential link between two nodes is predicted. The following experiments were carried out:

- Experiment I: Comparison with the baseline VGAE model using configuration 1.
  - Target of Exp. I: to compare configuration 1 with the baseline results.
- Experiment II: Evaluation and comparison of multiple architectures and configuration 2 for the three benchmark datasets considered in this work.
  - Target of Exp. II: to compare configuration 2 with the baseline results.
- Experiment III: Timing comparison between both configurations for the three datasets.
  - Target of Exp. III: to compare the configurations 1 and 2 for the time of execution.
- Experiment IV: Timing comparison using central processing unit (CPU) and GPU for both configurations for the three datasets.
  - Target of Exp. IV: to compare and understand the usability and effectiveness of CPU and GPU on the basis of time of execution for both configurations.

All the experimentation was carried out using PyTorch 1.10 and PyTorch Geometric.[45,46] The initial experiments were carried out on a CPU. However, due to limitations on the processing capacity, the experiments were later conducted on a GPU using Google Colab. The outcome of the proposed framework would provide the probability of a possible link between two nodes of a network.

3.2 Parameters for configurations 1 and 2 models

Configuration 1 is designed specifically to compare the baseline results of VGAE (Table 2) and is somewhat minimalistic.[14] In contrast, configuration 2 is the proposed multi-head VGAEs, where experimentations with the hyperparameter tuning are carried out (refer to Table 2). This comparison provides the information regarding the effect of each parameter on the model training, its outcome, and the time required for processing.

3.3 Data preparation

The data preparation step involved pre-processing of the data. PyTorch-Geometric was used for automated data pre-processing. The three benchmark datasets were automatically downloaded in the variational autoencoder implementation using PyTorch Geometric. In the pre-processing step, the PyTorch geometric library loads the following parameters:

- Node features: The features associated with each node

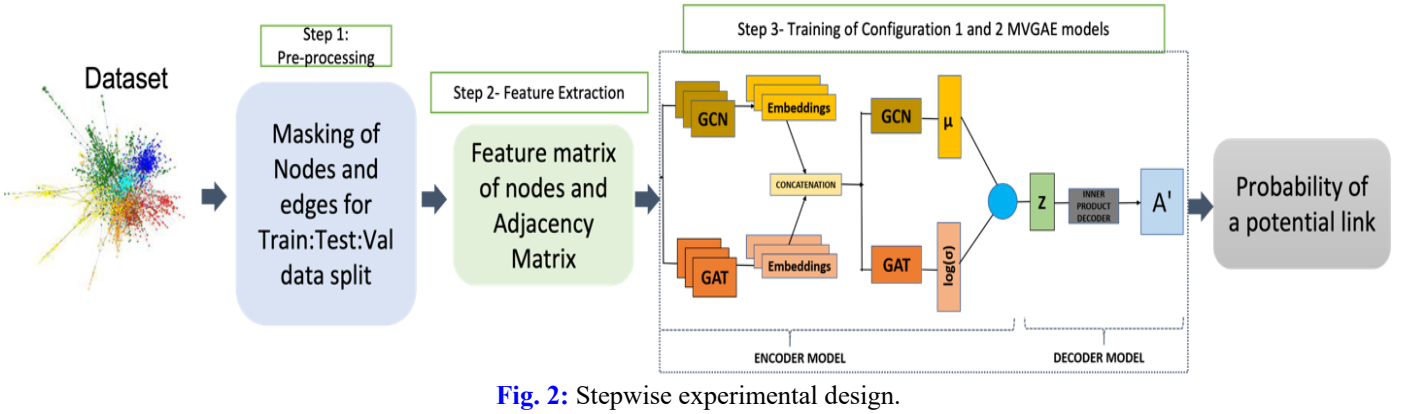

Fig. 2: Stepwise experimental design.

Table 2: Details of configurations 1 and 2 of the model.

Configuration name	1	2
Hidden dimension for layer 1	32	64
Hidden dimension for layer 2	16	32
Epochs	200	300
Learning rate	0.01	0.015375
Heads in the MVGAE used	GCNConv, GATConv	GCNConv, GATConv
Number of attention heads	2	4
Optimizer	Adaptive moment estimation (Adam)[47]	

typically represent attributes of the documents or papers. In Cora, Citeseer, and Pubmed datasets, these features may be bag-of-words vectors or term frequency-inverse document frequency (TF-IDF) features. PyTorch Geometric loads these features as node attributes.

- Graph Structure: The citation relationships between documents are represented as edges in the graph. PyTorch Geometric constructs the graph structure based on the citation information in the raw dataset.
- Creating automatic train, test, and validation split: Let's suppose we are using the Cora dataset, represented by Graph G. Before the creation of the split, masking of nodes is done. Masking refers to randomly switching off nodes and edges of the graph. Hence, three graphs are created for training, validation, and testing.

3.4 Comparison and evaluation metrics

Based on the baselines, the models are compared for two metrics, namely, the average precision (AP) score and the receiver operating characteristic (ROC) and area under curve (AUC) score. The train, validation, and test splits of the dataset are distributed as 85%, 5%, and 10%, respectively. As mentioned, the initial training was carried out using a CPU, followed by the GPU, using Google Colab for evaluating the scope for parallelization. The results obtained for all the experiments for AP and AUC are shown in the next subsections.

3.5 Results for experiment I

For comparison with the baseline VGAE model, configuration 1 of the model (Table 2) is employed, which has a similar configuration to the VGAE model.[14] Table 3 shows the model's outcomes in these three benchmark datasets using configuration 1.

Table 3: Comparison with the baseline model for configuration 1.

Data	Metric	GAE	VGAE	MVGAE
Cora	AP (%)	92.0	92.6	97.11
Citeseer	AP (%)	89.9	92.9	95.31
Pubmed	AP (%)	96.5	94.7	97.88
Cora	AUC (%)	91.0	91.4	97.28
Citeseer	AUC (%)	89.5	90.8	95.68
Pubmed	AUC (%)	96.4	94.4	98.18

Figs. 3-5 show the train loss curve, test AUC score curve, and test AP score curve for the model using configuration 1, respectively

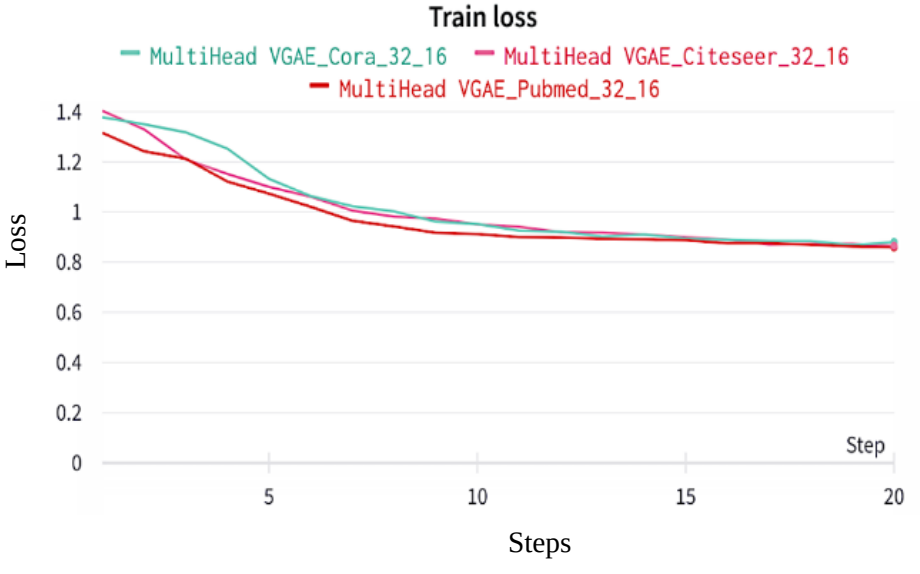

Fig. 3: Training loss of the base model.

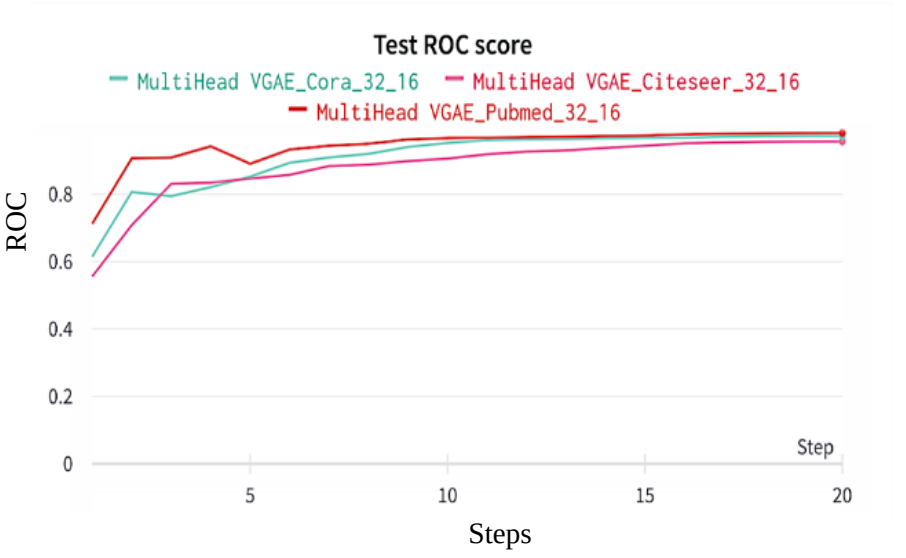

Fig. 4: Test ROC score from the base.

3.6 Results for experiment II

Recent state of the art models in link prediction include Walkpooling,[38] GNAE,[37] VGNAE,[37] S-VGAE,[48] sGraphite-VAE,[49] Graph InfoClust,[50] and ARGE.[51] Their comparison with the proposed MVGAE model are provided for all three datasets Cora, Citeseer, and Pubmed is provided in Tables 4-6 respectively. Configuration 2 (Table 2) is employed for this set of comparisons.

Table 4: Comparison of the proposed model with others on the Cora dataset.

SL. No	Model	AUC	AP
1	Walkpooling[38]	95.9	96.0
2	GNAE[37]	95.6	95.7
3	VGNAE[37]	95.4	95.8
4	S-VGAE[48]	94.1	94.1
5	sGraphite-VAE[49]	93.7	93.5
6	Graph InfoClust[50]	93.5	93.3
7	ARGE[51]	92.4	93.2
8	MVGAE (Proposed model)	99.1	99.2

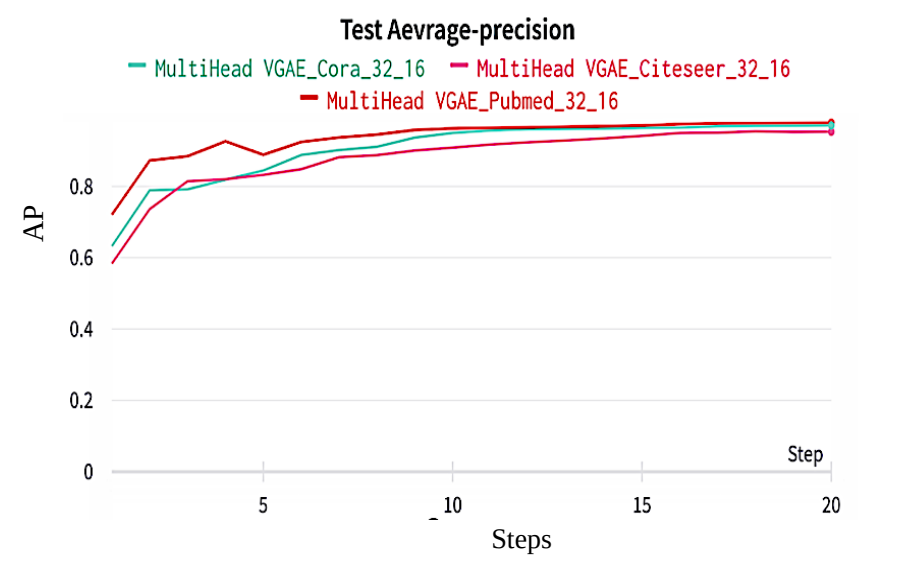

Fig. 5: Test AP score from the base model.

Table 5: Comparison of the proposed model with others on the Citeseer dataset.

SL. No	Model	AUC	AP
1	VGNAE[37]	97.0	97.1
2	Graph InfoClust[36]	97.0	96.8
3	GNAE[37]	96.5	97.0
4	Walkpooling[38]	95.9	96.0
5	sGraphite-VAE[49]	94.1	95.4
6	ARGE[50]	91.9	93.0
7	Node feature agg + similarity[51]	91.9	93.0
8	MVGAE (Proposed model)	98.5	98.3

Table 6: Comparison of the proposed model with others on the Pubmed dataset.

SL. No	Model	AUC	AP
1	Walkpooling[38]	98.7	98.7
2	GNAE[37]	97.6	97.6
3	VGNAE[37]	97.5	97.5
4	S-VGAE[45]	96.8	97.1
5	sGraphite-VAE [49]	96.0	96.0
6	Graph InfoClust [36]	94.8	96.3
7	ARGE[50]	93.7	93.5
8	MVGAE (Proposed model)	99.1	98.9

As can be observed from Tables 4-6, the proposed model has outperformed all other models in terms of the evaluation metrics of AUC and AP. Figs. 6-8 show the train Loss curve, test AUC score curve, and test AP score curve for the model using configuration 2. We can see from Tables 4-6 that our proposed model outperformed all the existing models.

3.7 Results for experiment III

Configuration 1 takes less time than configuration 2 as configuration 1 is simpler than configuration 2. Thus, for visualization, Figs. 9 and 10 show the time lapsed for training on the Cora, Citeseer, and Pubmed datasets per epoch and the total time lapsed to train them, respectively. Similarly, Fig. 11 shows the time lapsed for training using configuration 2.

3.8 Results for experiment IV

In this work, the initial experiments were carried out on CPU, and then we shifted to GPU, using Google Collab. It was observed that there is a drastic improvement in the time taken by the model to run. Table 7 shows the comparison of time taken by the CPU and GPU to run the model in configurations 1 and 2, respectively.

Table 7: Time required to train vs the number of VGAE heads.

Dataset	Configuration	Total time in CPU (minutes)	Total time in GPU (minutes)
Cora	1	34.3896	6.0645
Cora	2	80.3523	8.5429
Citeseer	1	52.0184	7.2918
Citeseer	2	119.9359	8.5624
Pubmed	1	454.3810	41.7666
Pubmed	2	1033.8588	45.1359

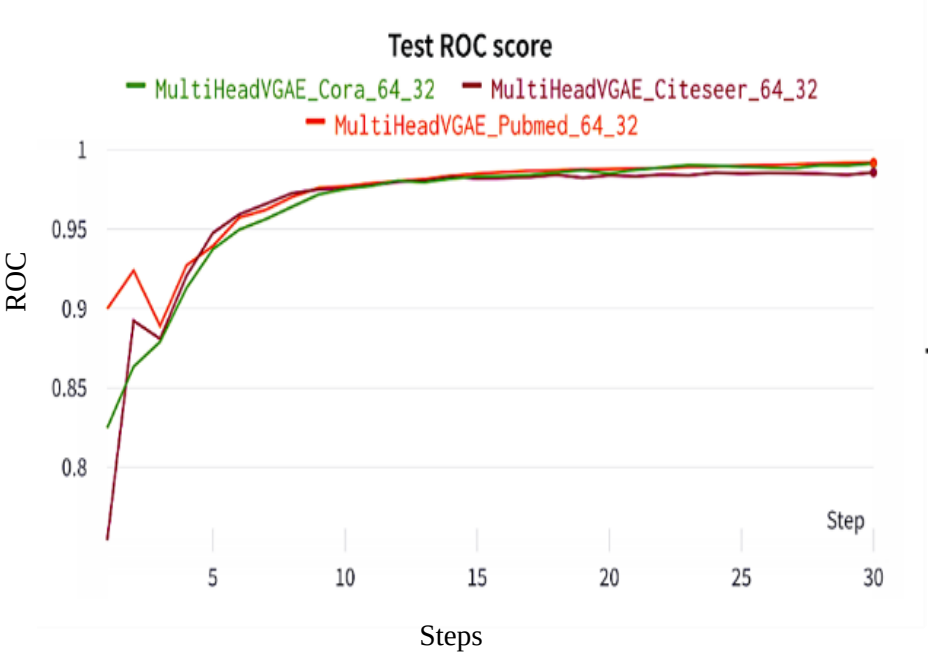

Fig. 6: Train loss from configuration 2.

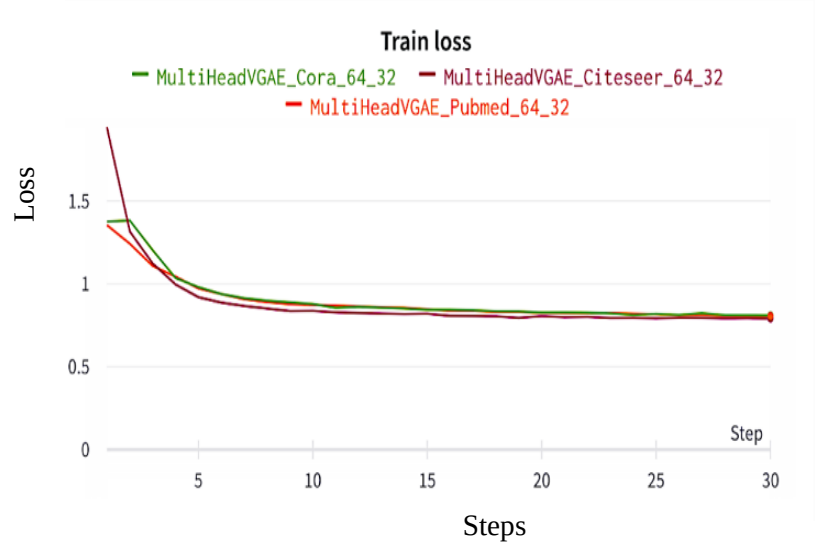

Fig. 7: Test ROC score for configuration 2.

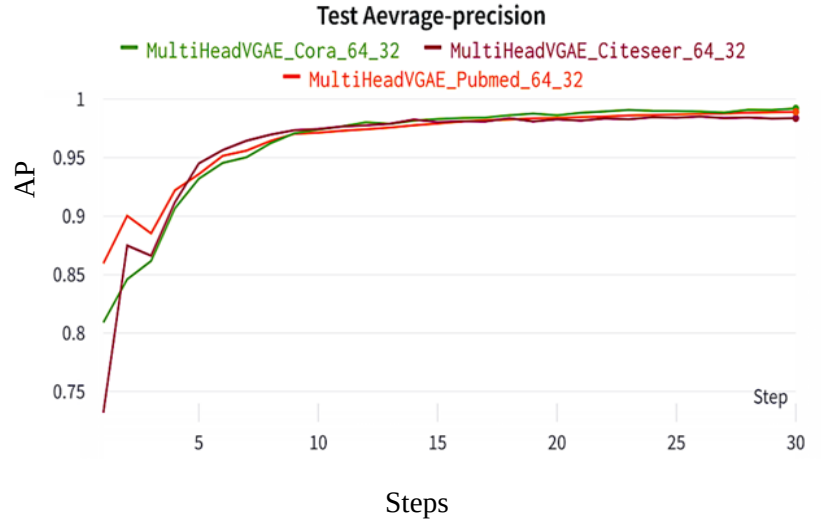

Fig. 8: Test AP score from the model configuration 2.

Fig. 9: Training time/epoch by the base model.

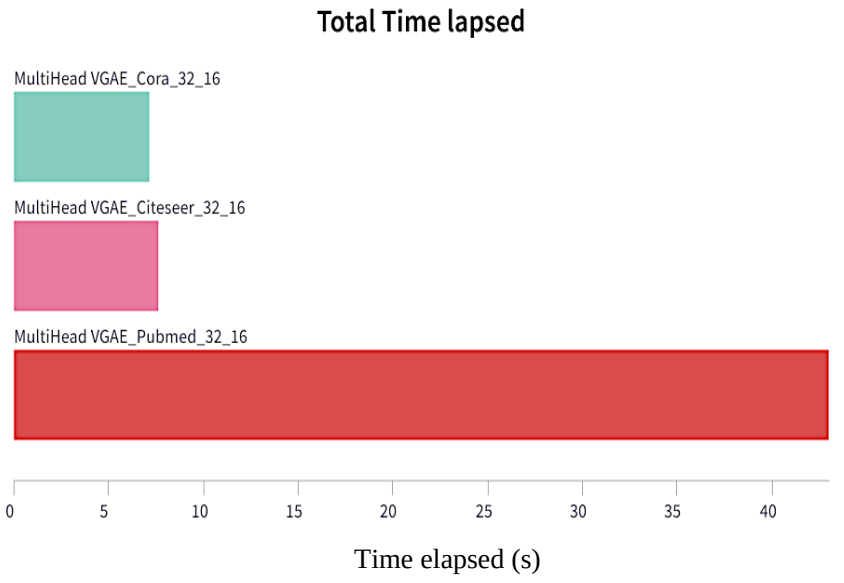

Fig. 10: Total training time by the base model.

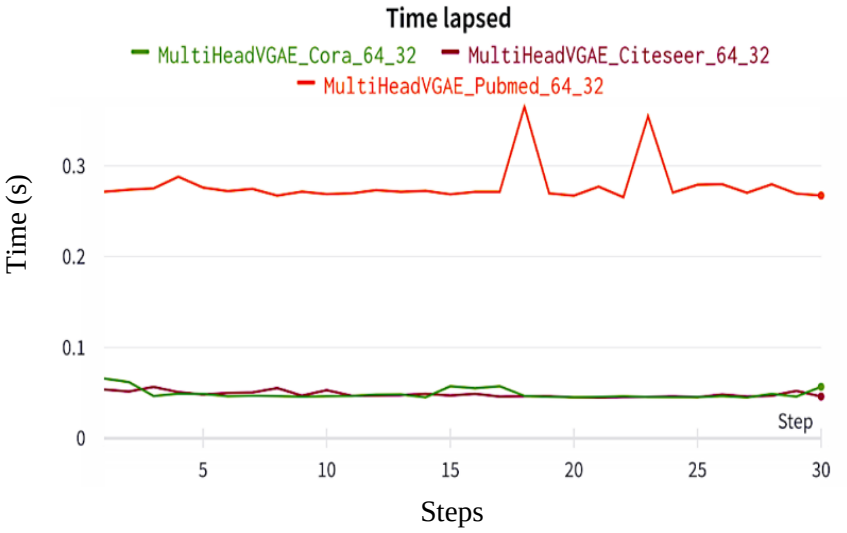

Fig. 11: Training time/epoch by model configuration 2.

4. Conclusion

This work presents an innovative and efficient architecture for link prediction tasks in citation-based networks. By introducing multiple parallel heads of different layers and aggregating the unified hidden representations, the proposed MultiHead VGAEs model surpasses existing state-of-the-art models in terms of computational performance and evaluation metrics. The experiments conducted demonstrate the superior performance of the proposed method in link prediction tasks within citation-based networks. The utilization of GPU acceleration significantly enhances the model's performance and reduces training time, as evidenced by the substantial leap in efficiency observed when transitioning from configuration 1 to configuration 2, particularly in the Pubmed dataset. This highlights the importance of leveraging hardware acceleration for optimal performance. While this work focuses on link prediction in citation-based graphs, the proposed architecture has the potential for broader applicability across different types of graphs, including user-item bipartite graphs.

Future research can explore extending the architecture to address generalized link prediction tasks in various graph domains. However, it is important to acknowledge the limitations of the proposed model. Being a transductive architecture, it may not be suitable for scenarios where the graph structure evolves or when predicting links in unseen nodes. Additionally, it is crucial to consider the novelty and simplicity of the experimental setup and the potential improvements it offers in real-world link prediction use cases. In future endeavors, it would be beneficial to explore architectural enhancements such as incorporating skip connections or custom decoder layers to enhance the model's capabilities further. Additionally, investigating potential

setbacks and limitations of the proposed architecture will contribute to a comprehensive understanding of its applicability.

Overall, this work presents a compelling approach to link prediction in citation-based networks, showcasing the effectiveness of multi-head VGAEs. By combining novel architectural design, efficient computational performance, and promising evaluation results, this research contributes to the advancement of graph representation learning and holds promise for a wide range of applications in various domains. This approach, although presented here for the link prediction task, can be extended further for other graph modelling methods and tasks.

Conflict of Interest

There is no conflict of interest.

Supporting Information

Not applicable.

References

[1] Y. Zhang, L. Luo, W. Xian, H. Huang, Learning better visual data similarities via new grouplet non-euclidean embedding, 2021 IEEE/CVF International Conference on Computer Vision, October 10-17, Montreal, QC, Canada, IEEE, 2021, 9898-9907, doi: 10.1109/iccv48922.2021.00977.

[2] H. Cai, V. W. Zheng, K. C. Chang, A comprehensive survey of graph embedding: problems, techniques, and applications, *IEEE Transactions on Knowledge and Data Engineering*, 2018, **30**, 1616-1637, doi: 10.1109/TKDE.2018.2807452.

[3] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: Towards deep graph convolutional networks on node classification, arxiv preprint arxiv: 1907.10903, 2019, doi: 10.48550/arXiv.1907.10903.

[4] N. de Lara, E. Pineau, A simple baseline algorithm for graph classification, arxiv preprint arxiv: 1810.09155, 2018, doi: 10.48550/arXiv.1810.09155.

[5] V. Martínez, F. Berzal, J. C. Cubero, A survey of link prediction in complex networks, *ACM Computing Surveys*, 2017, **49**, 1-33, doi: 10.1145/3012704.

[6] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2018, 5171-5181, doi: 10.48550/arXiv.1802.09691.

[7] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: online learning of social representations, Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA, 2014, doi: 10.1145/2623330.2623732.

[8] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 2016, doi: 10.1145/2939672.2939754.

[9] Y. Dong, N. V. Chawla, A. Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax NS Canada, ACM, 2017, doi: 10.1145/3097983.3098036.

[10] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, arxiv preprint arxiv: 1706.02216, 2017, doi: 10.48550/arXiv.1706.02216.

[11] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, Proceedings of the Fifth International Conference on Learning Representations, 2017, doi: 10.48550/arXiv.1609.02907.

[12] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arxiv: 1710.10903, 2017, doi: 10.48550/arXiv.1710.10903.

[13] P. Barceló, E. Kostylev, M. Monet, I. Pérez, J. Reutter, J. P. Silva, The logical expressiveness of graph neural networks, In the 8th International Conference on Learning Representations, Addis Ababa, Ethopia, April 26-30,2020.

[14] T. N. Kipf, M. Welling, Variational graph auto-encoders, arxiv preprint arxiv: 1611.07308, 2016, doi: 10.48550/arXiv.1611.07308.

[15] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, *AI Open*, 2020, **1**, 57-81, doi: 10.1016/j.aiopen.2021.01.001.

[16] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Computation*, 2003, **15**, 1373-1396, doi: 10.1162/089976603321780317.

[17] H. Ma, H. Yang, M. R. Lyu, I. King, SoRec: Social recommendation using probabilistic matrix factorization, Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, California, USA, 2008, doi: 10.1145/1458082.1458205.

[18] C. Wang, Q. Liu, R. Wu, E. Chen, C. Liu, X. Huang, Z. Huang, Confidence-aware matrix factorization for recommender systems, *AAAI Conference on Artificial Intelligence*, 2018, **32**, 434-442, doi: 10.1609/aaai.v32i1.11251.

[19] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, L. Sun, Dual-regularized matrix factorization with deep neural networks for recommender systems, *Knowledge-Based Systems*, 2018, **145**, 46-58, doi: 10.1016/j.knosys.2018.01.003.

[20] Y. LeCun, P. Haffner, L. Bottou, Y. Bengio, Object recognition with gradient-based learning, Shape, Contour and Grouping in Computer Vision, Springer, Berlin, Heidelberg, 1999, **1681**, 319-345, ISBN: 978-3-540-66722-3.

[21] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arxiv preprint arxiv: 1412.3555, 2014, doi: 10.48550/arXiv.1412.3555.

[22] K. Yu, S. Zhu, J. Lafferty, Y. Gong, Fast nonparametric matrix factorization for large-scale collaborative filtering, Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston,

MA, USA, 2009, 211-218, doi: 10.1145/1571941.1571979.
[23] N. Wang, T. Yao, J. Wang, D. Yeung, A probabilistic approach to robust matrix factorization, Computer Vision - ECCV 2012, Springer, Berlin, Heidelberg, 2012, **7578**, 126-139, doi: 10.1007/978-3-642-33786-4_10.
[24] Y. Zhou, C. Wu, L. Tan, Biased random walk with restart for link prediction with graph embedding method, *Physica A: Statistical Mechanics and its Applications*, 2021, **570**, 125783, doi: 10.1016/j.physa.2021.125783.
[25] X. Du, J. Yan, R. Zhang, H. Zha, Cross-network skip-gram embedding for joint network alignment and link prediction, *IEEE Transactions on Knowledge and Data Engineering*, 2020, **34**, 1080-1095, doi: 10.1109/tkde.2020.2997861.
[26] K. Berahmand, E. Nasiri, M. Rostami, S. Forouzandeh, A modified DeepWalk method for link prediction in attributed social network, *Computing*, 2021, **103**, 2227-2249, doi: 10.1007/s00607-021-00982-2.
[27] J. Chen, X. Lin, Z. Shi, Y. Liu, Link prediction adversarial attack *via* iterative gradient attack, *IEEE Transactions on Computational Social Systems*, 2020, **7**, 1081-1094, doi: 10.1109/tcss.2020.3004059.
[28] H. Chen, H. Yin, X. Sun, T. Chen, B. Gabrys, K. Musial, Multi-level graph convolutional networks for cross-platform anchor link prediction, Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 2020, doi: 10.1145/3394486.3403201.
[29] M. Coşkun, M. Koyutürk, Node similarity-based graph convolution for link prediction in biological networks, *Bioinformatics*, 2021, **37**, 4501-4508, doi: 10.1093/bioinformatics/btab464.
[30] J. Chen, X. Wang, X. Xu, GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction, *Applied Intelligence*, 2022, **52**, 7513-7528, doi: 10.1007/s10489-021-02518-9.
[31] A. Graves, N. Jaitly, A. R. Mohamed, Hybrid speech recognition with deep bidirectional LSTM, 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, December 8-12, Olomouc, Czech Republic, IEEE, 2013, 273-278, doi: 10.1109/ASRU.2013.6707742.
[32] K. Lei, M. Qin, B. Bai, G. Zhang, M. Yang, GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications April 29-May 2, Paris, France, IEEE, 2019, doi: 10.1109/infocom.2019.8737631.
[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A Courville, Y. Bengio, Generative adversarial networks, *Communications of the ACM*, 2020, **63**, 139-144, doi: 10.48550/arXiv.1406.2661.
[34] Z. Yang, W. W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, 33rd International Conference on Machine Learning, 2016, 40-48, doi: 10.48550/arXiv.1603.08861.
[35] P. V. Tran, Multi-task graph autoencoders, arxiv preprint arxiv: 1811.02798, 2018, doi: 10.48550/arXiv.1811.02798.
[36] C. Mavromatis, G. Karypis, Graph InfoClust: Leveraging cluster-level node information for unsupervised graph representation learning, arxiv preprint arxiv: 2009.06946, 2020, doi: 10.48550/arXiv.2009.06946.
[37] S. J. Ahn, M. Kim, Variational graph normalized autoencoders, Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Virtual Event Queensland, Australia, 2021, doi: 10.1145/3459637.3482215.
[38] S. Yun, S. Kim, J. Lee, J. Kang, H. J. Kim, Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction, *Advances in Neural Information Processing Systems*, 2022, **34**, 13683-13694, doi: 10.48550/arXiv.2206.04216.
[39] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arxiv preprint arxiv: 1312.6203, 2013, doi: 10.48550/arXiv.1312.6203.
[40] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, 3844-3852, doi: 10.48550/arXiv.1606.09375.
[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, *Advances in Neural Information Processing Systems*, 2017, **30**, 1-15, doi: 10.48550/arXiv.1706.03762.
[42] D. P. Kingma, M. Welling, Auto-encoding variational Bayes, arxiv preprint arxiv: 1312.6114, 2013, doi: 10.48550/arXiv.1312.6114.
[43] J. Goldberger, S. Gordon, H. Greenspan, An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures, Proceedings 9th IEEE International Conference on Computer Vision, October 13-16, Nice, France, IEEE, 2003, **2**, 487-493, doi: 10.1109/ICCV.2003.1238387.
[44] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, 2017 IEEE Conference on Computer Vision and Pattern Recognition, July 21-26, Honolulu, HI, USA, IEEE, 2017, 5987-5995, doi: 10.1109/CVPR.2017.634.
[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, *Advances in Neural Information Processing Systems*, 2019, **32**, 1-12, doi: 10.48550/arXiv.1912.0170.
[46] M. Fey, J. E. Lenssen, Fast graph representation learning with PyTorch geometric, arxiv preprint arxiv: 1903.02428, 2019, doi: 10.48550/arXiv.1903.02428.
[47] D. P. Kingma, J. Ba, M. M. Hammad, Adam: A method for stochastic optimization, arxiv preprint arxiv: 1412.6980, 2014, doi: 10.48550/arXiv.1412.6980.
[48] X. Di, P. Yu, R. Bu, M. Sun, Mutual information maximization in graph neural networks, 2020 International Joint

Conference on Neural Networks, July 19-24, Glasgow, UK, IEEE, 2020, doi: 10.1109/ijcnn48605.2020.9207076.
[49] A. Grover, A. Zweig, S. Ermon, Graphite: Iterative generative modeling of graphs, International Conference on Machine Learning, 2019, 2434-2444, doi: 10.48550/arXiv.1803.10459.
[50] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, *IEEE Transactions on Cybernetics*, 2020, **50**, 2475-2487, doi: 10.1109/TCYB.2019.2932096.
[51] Y. Tian, L. Zhao, X. Peng, D. Metaxas, Rethinking kernel methods for node representation learning on graphs, *Advances in Neural Information Processing Systems*, 2019, **32**, 1-2, doi: 10.48550/arXiv.1910.02548.