# Summary

**Github Link:** [**https://github.com/tengann/IAV\_PPI\_Graph\_Embedding\_Review**](https://github.com/tengann/IAV_PPI_Graph_Embedding_Review)

* **Review for Human-IAV host-pathogen interactions,**

**based only on 4 strains of interest**

|  |  |  |  |
| --- | --- | --- | --- |
| **Strain** | **Subtype** | **Abbreviation** | **Reason** |
| A/Puerto Rico/8/1934 | H1N1 | PR8 | Commonly used in lab experiments |
| A/California/04/2009 | H1N1 | CA04 | 2009 Pandemic strain |
| A/California/07/2009 | H1N1 | CA07 | 2009 Pandemic strain  (Contains an “unusual” amino acid ‘X’ in its NP segment) |
| A/Aichi/2/1968 | H3N2 | Aichi | H3N2 subtypes of IAV have cause seasonal epidemics since 1968 [1] |

**Datasets:**

1. PPIDomainMiner (PPIDM) [2] Construct dataset based on 4 strains of interest

Dataset of 84, 552 non-redundant DDIs

Domain interactions, interactions do not differ between strains

Domain Analysis (Superfamily VS Pfam)??

1. HVPPI [3]

* Unsupervised sequence embedding technique (doc2vec) to represent protein sequences as rich feature vectors of low dimensionality.

Training a Random Forest (RF) classifier through a training dataset that covers known PPIs between human and all viruses

* Host-pathogen PPI data from the **Host-Pathogen Interaction Database (HPIDB)** [4, 5] V3.0
  + Contains manually curated host-pathogen interactions and also integrates corresponding molecular interactions from other public protein interaction databases
* Built based on an unbalanced human host-virus PPI dataset with positive-to-negative ratio 1:10

Performance of HVPPI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Accuracy (%) | Sensitivity (%) | Specificity (%) | F1-Score (%) | ROC-AUC |
| 79.17 | 81.85 | 76.45 | 79.79 | 0.871 |

[Show comparison of the distribution of interacting pairs across segments]

**Models:**

1. **Feedforward Neural Network (FNN)**

Sequential model (i.e., plain stack of layers where each layer has exactly one input tensor and one output tensor) (keras)

Same configuration as classifier used in S-VGAE [6]

3-layers: input + 2 hidden, with 128, 64 and 32 neurons in each layer

Uses dropout technique, important trick widely used in deep learning [7]

Train for 200 epochs using Adam algorithm with a batch size of 128

**[Performance of classifier can be remarkably good without complex neural network structures, since the embeddings already contain enough information and are highly representative in the learned low-dimensional vector space] [6]**

1. **Skip-GNN (pytorch) [8]**

**FNN**

* First and simplest type of artificial neural network devised [9]
* Connections between nodes do not form a cycles or loops in the network. Information moves in only one direction. [10]

**Skip-GNN**

* Leverage accessible network information (adjacent matrix of the network G) to predict links
* Skip similarity (similarity in second-order proximity interactions, from second-hop neighbors) by encouraging the GNN model to embed skipped nodes close together in the embedding space
* Construct skip graph from second-hop neighbours

Element-wise matrix multiplication

A­s = sign(AAT), where sign(AAT) = 1 if AAT > 0, and sign(AAT) = 0 otherwise, where AAT= 1 indicates there exists a skipped node between nodei, j

* Iterative fusion scheme (instead of simple concatenation) to allow the skip graph and original graph to learn from each other for better integration. Model automically learn how to balance between direct similarity and skip similarity in the final embedding.

Skip-GNN and S-VGAE investigate different aspects

Skip-GNN (similarity in second-order interactions, from second-hop neighbors)

* Not quite affected by chosen embedding algorithm

S-VGAE (signed adjacency matrix) – set different signs to reinforce existing observed interactions (strengthen the negative impact of highly negative interactions)

(if works better than VAE)

**Graph embedding methods: [Only making use of the graph network topology]**

|  |  |
| --- | --- |
| Method Category | Method Name |
| Matrix factorization-based | GraRep [11] |
| Random walk-based | Deepwalk [12]  node2vec [13]  struc2vec [14] |
| Neural network-based | LINE [15]  SDNE [16]  VAE [17] |

Used to calculate *static* graph embeddings (i.e., computed for a graph at a fixed time period)

<https://towardsdatascience.com/lets-talk-about-graph-machine-learning-in-biomedical-networks-8a84139e970b>

Codes:

* Deepwalk, node2vec, struc2vec

<https://github.com/shenweichen/GraphEmbedding> (Repo used in Skip-GNN paper)

* LINE, SDNE, GraRep, VAE

<https://github.com/xiangyue9607/BioNEV>

* Other algorithms:

Priority:

* **Node2Vec+ [18]**,implemented as part of **pecanpy** [19]

Natural extension of node2vec and handles weighted graph more effectively

<https://github.com/krishnanlab/PecanPy>

SparseOTF vs DenseOTF

* **ripple2vec [20]** – recent methodology

Node Embedding with Ripple Distance of Structures

<https://github.com/hitSongXiao/ripple2vec>

Implemented in the framework of struc2vec by adapting ripple distance to define context graphs

Helps map dis(similar) nodes to (far) near vectors

[Use same hyperparameters (num walks & walk length) as in struc2vec]

* **MixHop –** Higher order convolutional layer

Designed to capture higher order graph structure through skip connections and higher order adjacency matrix

Motivated by general network model and does not propose a solution for the specific challenge of 2-hop skip similarity in biomedical network [8]

<https://github.com/benedekrozemberczki/MixHop-and-N-GCN>

[Skip-GNN used MixHop for comparison]

GCN VS Skip-GNN VS GAT

Others:

* **Edge2vec [21]**

(To investigate the effect of edge weights)

<https://github.com/RoyZhengGao/edge2vec>

Designed for heterogeneous information networks, containing diverse biological entities (e.g., genes, proteins, drugs, phenotypes)

* **Attre2vec**

https://github.com/attre2vec/attre2vec

**Contributions: Points explored/to explore**

1. **With VS without edge weights**

Weights on edges are useful for graph representational learning [22]

However, PPI networks are modelled as undirected unweighted graphs

(Use PPIDM - statistical significance (p-value) and

HVPPI - interaction probability of query protein pair as weights)

Using HVPPI prediction score to learn node features for **non-edges**

(Assigned weights to non-edges instead of simply **zero**)

*No experimentally verified data for non-interacting Human-IAV protein pairs. Therefore, may still have a slight chance of interaction?*

**Currently, HVPPI probability scores are helpful in choosing edges for constructing the dataset but not leveraged when learning graph representations**

**[Ref Github: results/Choosing\_HVPPI\_negative\_threshold.xlsx]**

* Reduce noise [23]

If node2vec+ works, otherwise just mention the above. And mention limitations with current algorithms?

1. **With VS without hyper-parameter tuning** (training of graph embeddings)

Reference to paper [23]

Compare min and max results to show how much results can differ

Paper [23] did not mention to optimize number and length of walks for **node2vec**

However, author of node2vec paper [13] mentioned that both these parameters have a relatively high impact on the performance of the method.

**Node2vec**: [Use same hyperparameters (num walks & walk length) as in deepwalk] – to make comparison of Deepwalk VS node2vec

* **HVPPI predictions dataset** – update results for node2vec (num walks & walk length)
* **Experimentally verified dataset** – Top algorithm for both FNN (SDNE) & Skip-GNN [Choose the appropriate hyper-parameters] – Can achieve better performance than protein embedding

1. **Graph embedding (Individual VS Ensemble of methods) ??**

Finding of the best performing method

[Combination of Top 3 embedding methods & All 6 methods]

1. **Comparison to results obtained with protein embedding**

Can only be tested on FNN, as Skip-GNN is designed to only leverage graph network topology information

[Link prediction on experimentally verified dataset]

Methods mentioned in DeepPPI [24] paper

* + Protein embedding methods, iFeature [25]
* Amino acid composition (AAC)
* Dipeptide Composition (DPC)
* Composition, Transition and Distribution (C/T/D)
* Quasi-Sequence-Order Descriptors (QSOrder)
* Amphiphilic Pseudoamino Acid Composition (APAAC)
* Conjoint Triad (CT)
* Normalized Moreau-Broto Autocorrelation (NMBroto)

1. **Concatenating graph and protein embedding**

Reviewing methodology proposed in paper [7]

[Best performing graph + best performing protein]

(points 4 & 5 – embeddings retrieved but yet to be tested)

S-VGAE (signed adjacency matrix) vs VAE

1. **Computation of edge embeddings (representation for pairs of nodes)**

Compare between Concatenation, Sum, Average and Hadamard Product

(sum and hadamard product – keep original dimensions)

*S-VGAE* – hstack (equivalent to concatenation)

*SkipGNN* – “concatenation consistently yield the best performance across different types of networks”

[Using optimized hyper-parameters]

(Briefly explored on Feedforward Neural Network,

From my results, hadamard product tends to achieve better performance instead of concatenation)

1. Performance on unbalanced dataset? – additional

(i.e., different proportions of positive:negative samples) [26]

Currently, ratio of positive:negative is 1:1

Comparison of PR-ROC scores

[Can show results based on top graph (& protein) embedding methods on both Skip-GNN and Feedforward NN classifier] – show reliability of graph embeddings

[Use same hyper-parameters as in balanced dataset]

1. Analysis on graph constructed by HVPPI

[Network properties? (e.g. degree, clusters)]

# Section 1: Introduction

1. **Significance of review**

There are still limited works that use graph embedding in bioinformatics, where most use either protein sequence or structure instead

* High research output on Graph Machine Learning, especially Graph Neural Networks (GNN) – formulating different tasks on biomedical graphs enables us to tackle some *bottlenecks* of the traditional lab experiments
* Computational link prediction: Has a high-throughput, however, often have high false positive and negative rates

**Requires a “strong” conclusion**

* **Demonstrate that graph topology embedding can achieve competitive results relative to protein sequences, using a specific problem (human-IAV PPI)**
* **Components of graph network topology that is important**

**e.g. Hops? Higher-order proximity?**

First-order proximity: Pairwise proximity (distance?) between vertices (Vi and Vj)

Second-order proximity: Similarity between neighborhood structure (NVi and NVj) of vertices Vi and Vj [27]

Skip-GNN: Considers the similarity between neighbors of (NVi and NVj) – “Third(Higher)-order proximity”

* **Do we really need a complex classification model or just high-quality embeddings?**

1. **Pros of using graph embedding**

* Simple and low-computational complexity
* Biological features typically faces two problems: [23]

1. May not always be available and can be hard and costly to obtain

Removing biological entities without features via pre-processing usually results in small-scale pruned datasets and thus is not pragmatic and useful in the real setting

1. Biological features & hand-crafted graph features (e.g. node degrees), may not be precise enough to represent or characterize biomedical entities

May fail to help build a robust and accurate model for many applications

[28]

* Unusual (non-standard) amino acid residues (e.g. ‘X’) cannot be encoded

1. **FNN (relatively simpler model) VS Skip-GNN**

However, for Skip-GNN, results do not vary much between embedding methods.

(Possible reason is prediction is largely dependent on the skip graph instead of the original graph)

SDNE performs well on FNN when α=0 indicates that performance totally determined by the second-order proximity

1. **Past works**

Examples of past works that used protein sequence/structure information to initialize node attributes (i.e., protein sequences/structure information are projected into a homogeneous vector space)

* **HVPPI** [3] uses doc2vec, which requires the protein sequence
* **DeepPPI** [24] (fuse multiple protein feature extraction methods)??
* **DPPI** [29] uses PSI-BLAST, derives a position-specific scoring matrix (PSSM)
* **Struct2Graph** [30]– GCN with mutual attention (requires PDB structures)

Protein sequence + Graph network topology

* **S-VGAE** [6] - Improved graph representation learning method, to incorporate graph information in PPI networks into PPI prediction. (Encoder-Decoder approach)

Abstract features are based on both sequence information and graph structure

Uses sequence representations by the conjoint triad (CT) method as input features of each node

Setting weights to edges using signed adjacency matrix, instead of only assigning 0 and 1 to the negative and positive group respectively, highly negative group assigned -1 and uncertain group assigned 0

(How was the highly negative group determined?)

* **Paper [7] (Liu et al)** – proposed a representation method that combined amino acid sequence information and position information to generate a stronger (node) representation for the protein (direct concatenation)

Uses one-hot to encode protein sequence, GCN to capture position information

* **Topsy-Turvy [31]**

Sequence-only model D-SCRIPT + Network-only model: GLIDE

D-SCRIPT – Protein language model + CNN (bottom-up approach)

GLIDE – scores all possible edges using a weighted combination of global and local network scores which are learned from the edges already in the training network

(top-down approach)

Examples of past works that used only graph embedding to initialize node attributes

* **HO-VGAE** [32] - Graph embedding-based via higher-order GCN.

Combined GCN with a personalized PageRank algorithm (variant of random walk)

Connects the random walk-based propagation effect (scheme) of personalized PageRank to GCN consecutively in every convolutional layer

* **SkipGNN** [8] – only leverage accessible network information (adjacent matrix A of the network G) to predict links. In all experiments, only **node2vec** was used to initialize the node attributes.

# Section 2: Overview of graph embedding methods

**[Briefly describe algorithm/implementation of each method]**

## Key Terms

* First-order proximity

Connected nodes in a graph should have similar properties

* Second-order proximity

Nodes with similar neighborhoods should have common characteristics

1. **Random walk-based**

Random walks are computationally efficient in terms of both space and time requirements [13]

* Deepwalk

Learns structural regularities present within short truncated random walks [12]

* Node2vec

Flexible biased random walk procedure that can smoothly interpolate between Breadth-first Sampling (BFS) and Depth-first Sampling (DFS) to explore neighborhoods [13]

Introduces two parameters [p (in-out) and q (return)] to control the graph structure with sampling, which import external parameters to search the structure of graph with BFS and DFS rather than the original weights in the graph [22].

* Struc2vec

First constructs a multi-layer weighted graph to generate context for each node.

Biased random walk is then performed on the multilayer graph to learn node sequences.

These sequences are likely to include nodes that are more structurally similar [14].

Applies Dynamic Time Warping (DTW) [33] on degree sequences to measure the similarity of local structures [20]. Nodes with high structural similarity are close to each other in the embedding space [23].

DTW makes distance comparisons less sensitive to signal transformations as shifting, uniform amplitude scaling or uniform time scaling [34].

However, DTW algorithm ignore partially the effects of connection patterns within neighborhoods. As a result, nodes with similar local structures may be mapped to far vectors [20]. (reason for comparison to ripple2vec)

1. **Neural network-based**

* LINE

Has a carefully designed objective function, optimized by an edge-sampling algorithm, that preserves both first-order and second-order proximities [15]

* SDNE

Semi-supervised [16] deep autoencoder along with Laplacian Eigenmaps (Matrix factorization-based) [35], which simultaneously optimizes the first-order and second-order proximity. The learned representations preserve the local and global network structure.

Laplacian Eigenmaps [36] generates network representations by factorizing the Laplacian matrix of the adjacency matrix. It exploits only the first-order proximity to preserve network structure [16].

* VAE

[Highlight its difference from GAE] – Purpose of variational

Utilizes a GCN encoder and an inner product decoder to learn node embeddings [23]

Can naturally incorporate node features [17] (e.g. protein sequence embedding)

Have “data-cleansing power”

Autoencoders distill inputs into the densest amount of data necessary to re-create a similar output

Removes data noise, transformed raw files into clean machine learning data and detect anomalies

<https://www.techtarget.com/searchenterpriseai/feature/How-to-troubleshoot-8-common-autoencoder-limitations>

Autoencoders are lossy – contain loss by aggressively pruning the problem space

1. **Matrix factorization-based**

Similarity-based learning – assumption that entities with similar interaction patterns are likely to interact

* GraRep

Extends to high-order proximity and uses the Singular Value Decomposition (SVD) to train the model [16].

Generalizes LINE to incorporate information from network neighborhoods beyond 2-hops, but is unable to efficiently scale to large networks [13].

**[Effects of hyper-parameters]** – Mentioned in [23]

1. **Random walk-based**

* **Node2vec**

Sampling strategy in Deepwalk can be seen as a special case of node2vec with p=1 and q=1 [13]

1. **Neural network-based**

* **SDNE**
* Alpha (α)

α = 0, performance totally determined by the second-order proximity

As α increases, it indicates that model is concentrating more on the first-order proximity

* Beta (β)

1. **Matrix factorization-based**

# Section 3: Experiments

Settings (standard hyper-parameters)

*Embedding dimensions = 128 (default given by the repo) – except VAE, depends on hidden dimensions*

*All results presented are* ***mean*** *of 5-fold cross validation*

Network properties – sparse network

(Proportion of positive (interacting) to negative (non-interacting) protein pairs in real world)

**HVPPI**

The same training set was used for all experiments, and two hold-out testsets were constructed

Number of nodes = 15, 685 (42 IAV, 15, 643 human proteins)

Total number of edges = 641, 404

However, as hvppi outputs predicted interactions, hvppi was ran twice and only edges that gave the same score based on 2 runs were considered

272, 091 edges, with 48,882 interacting, (~18% of all edges)

with 18, 654 above the 0.212 threshold

Randomly sampled 4460 positive

1. Based on predicted interactions by HVPPI

**[Network reconstruction]** Existing links in the original network are known and can preserve the original network structure

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No. of Samples** | **HVPPI Thresholds** | |
|  | **Training (80%) +  Validation (10%)** | **Min** | **Max** |
| Positive | 4014 | 0.212 | 0.99 |
| Negative | 4014 | 0 | 0.142 |
|  |  |  |  |
|  | **No. of Samples** | **HVPPI Thresholds** | |
|  | **Hold-out Test (10%)** | **Min** | **Max** |
| Positive | 446 | 0.212 | 0.931 |
| Negative | 446 | 0.001 | 0.141 |

**[Link prediction] (??)** Randomly hide a portion of the existing links and use the remaining network to train the network embedding methods

* Testset randomly sampled from the hidden portion (pairs that produced different scores on different hvppi runs)
* (did not choose threshold to allow more reliable results)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No. of Samples** | **HVPPI Thresholds** | |
|  | **Hold-out Test (10%)** | **Min** | **Max** |
| Positive | 446 | 0.144 | 0.591 |
| Negative | 446 | 0.009 | 0.142 |

FNN vs Skip-GNN, Top 3 methods

1. Experimentally verified datasets

HPIDB 3.0 (Positive) [4, 5] – Not in edgelist

<https://hpidb.igbb.msstate.edu/hpi.html>

Negatome 2.0 [37] (Negative)

Negatome - negative human-human protein interactions

Performance on dataset constructed based on HVPPI predictions VS experimentally verified datasets (HPIDB and Negatome)

Explain why use only human-human PPIs for negative

**PPIDM**

10,420 protein pairs, formed by 354 domain pairs (240 bronze, 114 silver)

4900 protein pairs, formed by 114 **silver** domain pairs

Negative (Randomly sampled from complement of positive dataset)

Positive train: 4279 protein pairs

1. with VS without edge weights

node2vec VS node2vec+

1. with VS without hyper-param tuning (effects of hyper-param tuning)

[training of graph embeddings]

With reference to paper [23]

1. Graph embedding

First-order proximity VS First + Second order proximity

1. with VS without protein embedding
2. Link prediction requires computation of representation for pairs of nodes

Edge embeddings – compare between Concatenation, Sum, Hadamard Product

(sum and hadamard product – keep original dimensions)

[Plot embeddings? + Results]

1. Analysis on graph network properties? (e.g. degree, clustering) [Aid in discussion of best-performing method?]

# Section 4: Discussion

**[Highlight Pros & Cons of each method]**

**Deepwalk**

* Does not have an explicit objective function to capture the network structure
* Uses random walk to enrich the vertexes, which introduces a lot of noises due to the randomness, especially for vertexes which have high degrees [16]

**Node2vec**

**Struc2vec**

* Determines a structural similarity between two nodes without any node or edge attributes [14]

**LINE**

* Adopts shallow structure, which is difficult to capture the highly non-linear structure in the underlying network [16]
* After optimizing the loss functions (defined separately to preserve the first-order and second-order proximity), LINE directly concatenates the representations, which is sub-optimal than jointly optimizing them in SDNE [16]

**SDNE [16]**

* Capture highly non-linear network structure

Semi-supervised deep model with multiple layers of non-linear functions

Address structure-preserving and sparsity problem

* Jointly optimize local and global network structure, learnt representations are local-global structure preserved and robust to sparse networks

**GraRep**

* Considers the high-order proximity of the network [16, 23]
* Directly concatenates the representations of first-order and high-order [16]

**SDNE best performing graph embedding algorithm on our dataset**

**Why VAE performance is bad compared to others (without features?)**

**[Discussion points]**

1. Skip-GNN VS FNN

Why choice of embedding algorithm does not affect Skip-GNN much (original graph)?

1. Edge weights

**Deepwalk** – only concerns that whether there is a connection between two nodes

**Node2vec**

**LINE** – mentioned that proposed model is able to apply to weighted graph, however, does not explicitly leverage the weights on graph

1. Hyper-parameter tuning

Graph embeddings are sensitive to hyperparameters

(Dependent on individual graphs)

Whereas, hyper-param tuning is not required for protein embedding

1. Ensemble of graph embeddings

Too large number of dimensions may introduce noises

e.g. performance drops when other methods added to SDNE

1. Comparison with protein embeddings

### Limitations

* HVPPI was trained solely on human-hosts and it is a challenge to find sufficient training data for other hosts
* For experimentally verified dataset, human-human only

Different from IAV-human? – High specificity and precision (i.e., low number of false positives) 0 identified as 1

# Section 5: Future work

Expansion to other strains

# References

1. Kakkola, L., N. Ikonen, and I. Julkunen, *Influenza A Viruses (Orthomyxoviridae)*, in *Encyclopedia of Virology (Fourth Edition)*, D.H. Bamford and M. Zuckerman, Editors. 2021, Academic Press: Oxford. p. 551-560.

2. Alborzi, S.Z., et al., *PPIDomainMiner: Inferring domain-domain interactions from multiple sources of protein-protein interactions.* PLoS Comput Biol, 2021. **17**(8): p. e1008844.

3. Yang, X., et al., *Prediction of human-virus protein-protein interactions through a sequence embedding-based machine learning method.* Comput Struct Biotechnol J, 2020. **18**: p. 153-161.

4. Kumar, R. and B. Nanduri, *HPIDB - a unified resource for host-pathogen interactions.* BMC Bioinformatics, 2010. **11**(6): p. S16.

5. Ammari, M.G., et al., *HPIDB 2.0: a curated database for host–pathogen interactions.* Database, 2016. **2016**: p. baw103.

6. Yang, F., et al., *Graph-based prediction of Protein-protein interactions with attributed signed graph embedding.* BMC Bioinformatics, 2020. **21**(1): p. 323.

7. Liu, L., et al., *Combining sequence and network information to enhance protein-protein interaction prediction.* BMC Bioinformatics, 2020. **21**(Suppl 16): p. 537.

8. Huang, K., et al., *SkipGNN: predicting molecular interactions with skip-graph networks.* Sci Rep, 2020. **10**(1): p. 21092.

9. Schmidhuber, J., *Deep learning in neural networks: An overview.* Neural Networks, 2015. **61**: p. 85-117.

10. Zell, A. *Simulation Neuronaler Netze [Simulation of Neural Networks] (in German)*. 1994.

11. Cao, S., W. Lu, and Q. Xu, *GraRep: Learning Graph Representations with Global Structural Information*, in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015. p. 891-900.

12. Perozzi, B., R. Al-Rfou, and S. Skiena, *DeepWalk: Online Learning of Social Representations.* Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014.

13. Grover, A. and J. Leskovec, *node2vec: Scalable Feature Learning for Networks.* KDD, 2016. **2016**: p. 855-864.

14. Ribeiro, L.F.R., P.H.P. Saverese, and D.R. Figueiredo, *struc2vec: Learning Node Representations from Structural Identity*, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017. p. 385-394.

15. Tang, J., et al., *LINE: Large-scale Information Network Embedding*, in *Proceedings of the 24th International Conference on World Wide Web*. 2015. p. 1067-1077.

16. Wang, D., P. Cui, and W. Zhu, *Structural Deep Network Embedding*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. p. 1225-1234.

17. Kipf, T.N.a.W., Max, *Variational Graph Auto-Encoders.* 2016.

18. Liu, R., M.J. Hirn, and A. Krishnan, *Accurately Modeling Biased Random Walks on Weighted Graphs Using Node2vec+.* CoRR, 2021. **abs/2109.08031**.

19. Liu, R. and A. Krishnan, *PecanPy: a fast, efficient and parallelized Python implementation of node2vec.* Bioinformatics, 2021. **37**(19): p. 3377-3379.

20. Luo, J., et al., *ripple2vec: Node Embedding with Ripple Distance of Structures.* Data Science and Engineering, 2022. **7**(2): p. 156-174.

21. Gao, Z., et al., *edge2vec: Representation learning using edge semantics for biomedical knowledge discovery.* BMC Bioinformatics, 2019. **20**(1): p. 306.

22. Wu, X., et al., *ProbWalk: A random walk approach in weighted graph embedding.* Procedia Computer Science, 2021. **183**: p. 683-689.

23. Yue, X., et al., *Graph embedding on biomedical networks: methods, applications and evaluations.* Bioinformatics, 2020. **36**(4): p. 1241-1251.

24. Du, X., et al., *DeepPPI: Boosting Prediction of Protein-Protein Interactions with Deep Neural Networks.* J Chem Inf Model, 2017. **57**(6): p. 1499-1510.

25. Chen, Z., et al., *iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences.* Bioinformatics, 2018. **34**(14): p. 2499-2502.

26. Dunham, B. and M.K. Ganapathiraju, *Benchmark Evaluation of Protein-Protein Interaction Prediction Algorithms.* Molecules, 2021. **27**(1).

27. Makarov, I., et al., *Survey on graph embeddings and their applications to machine learning problems on graphs.* PeerJ Comput Sci, 2021. **7**: p. e357.

28. Hamilton, W., R. Ying, and J. Leskovec, *Representation Learning on Graphs: Methods and Applications.* 2017.

29. Hashemifar, S., et al., *Predicting protein-protein interactions through sequence-based deep learning.* Bioinformatics, 2018. **34**(17): p. i802-i810.

30. Baranwal, M., et al., *Struct2Graph: a graph attention network for structure based predictions of protein–protein interactions.* BMC Bioinformatics, 2022. **23**(1): p. 370.

31. Singh, R., et al., *Topsy-Turvy: integrating a global view into sequence-based PPI prediction.* Bioinformatics, 2022. **38**(Supplement\_1): p. i264-i272.

32. Xiao, Z. and Y. Deng, *Graph embedding-based novel protein interaction prediction via higher-order graph convolutional network.* PLoS One, 2020. **15**(9): p. e0238915.

33. Keogh, E. and C.A. Ratanamahatana, *Exact indexing of dynamic time warping.* Knowledge and Information Systems, 2005. **7**(3): p. 358-386.

34. Cassisi, C., et al., *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. 2012.

35. Goyal, P., et al., *Embedding Networks with Edge Attributes*, in *Proceedings of the 29th on Hypertext and Social Media*. 2018. p. 38-42.

36. Belkin, M. and P. Niyogi, *Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.* Neural Computation, 2003. **15**(6): p. 1373-1396.

37. Blohm, P., et al., *Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis.* Nucleic Acids Res, 2014. **42**(Database issue): p. D396-400.