# Section 1: Introduction

1. **Significance of review**

* Currently, most works are evaluated only on benchmark datasets involving solely human-human PPIs. In this paper, Network Representation Learning (NRL), otherwise known as graph embedding methodologies are applied to a specific PPI problem (i.e., IAV-Human PPIs).
* There are still limited works that use graph embedding in bioinformatics, where most use either protein sequence or structure instead.
* Graph embedding methods are primarily evaluated on social and information (e.g., citation, coauthor) networks and not comprehensively studied on biomedical networks \cite{RN3}

1. **Pros of using NRL methods**

* Simple and low-computational complexity
* Biological features typically faces two problems: \cite{RN3}

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 \cite{RN31}

1. Biological features & hand-crafted graph features (e.g. node degrees), may not be precise enough to represent or characterize biomedical entities. Thus, may fail to help build a robust and accurate model for many applications \cite{RN31}

* Protein sequences that contain unusual (non-standard) amino acid residues (e.g. ‘X’) cannot be encoded. For example, in the case of NP segment of the CA07 strain, interactions with human proteins cannot be predicted.

1. **Review Points**

* Comparison of performance across various NRL methods
* Demonstrated that graph embedding is able to achieve competitive results relative to protein sequence embedding
* Conducted more detailed studies on the importance of preserving components of graph structure in their representations (i.e., second-order proximity (global structure) and edge attributes)
* Having high-quality embeddings proved to be more valuable compared to implementing a complex classification model

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** \cite{RN5} used doc2vec, which requires the protein sequence
* **DPPI** \cite{RN25} uses PSI-BLAST, derives a position-specific scoring matrix (PSSM)
* **Struct2Graph** \cite{RN27}– requires PDB structures, used GCN with mutual attention

Protein sequence + Graph network topology

* **S-VGAE** \cite{RN22}

Encoder-Decoder approach

Improved graph representation learning method, to incorporate graph information in PPI networks into PPI prediction.

Studied PPI prediction based on both sequence information and graph structure. Used conjoint triad (CT) method for protein sequence representation and signed adjacency matrix for graph embedding. Instead of only assigning 0 and 1 to the negative and positive group respectively, highly negative group assigned -1 (edge weight = -1) and uncertain group assigned 0 (unseen edges)

* **Paper \cite{RN12} by Liu et al**

Proposed a representation method that directly concatenated amino acid sequence information with position information to generate a stronger node representation for the protein. Used one-hot to encode protein sequence and GCN to capture position information

* **Topsy-Turvy \cite{RN47}**

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** \cite{RN28}

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** \cite{RN6}

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.

* Surveys of graph embedding methods on biomedical networks

\cite{RN3}

# Section 2: Overview of Network Representation Learning (NRL) methods

Codes:

* Deepwalk, node2vec, struc2vec

<https://github.com/shenweichen/GraphEmbedding>

* LINE, SDNE, GraRep, VAE

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

* nSNE

<https://github.com/wzsong17/Signed-Network-Embedding>

## Matrix factorization-based

Matrix factorization is a form of similarity-based learning, based on the assumption that entities with similar interaction patterns are likely to interact \cite{RN6}.

#### GraRep

* Preservers k-order proximity order \cite{RN18}.
* Accurately calculates the k-order proximity matrix \cite{RN62}.
* However, inefficient, especially when scaling to large networks or when k>= 2 \cite{RN62}.

## Random walk-based

Random walks are computationally efficient in terms of both space and time requirements \cite{RN7}

#### Deepwalk

* Learns structural regularities present within short truncated random walks \cite{RN21}

#### Node2vec

* Flexible biased random walk procedure that can smoothly interpolate between Breadth-first Sampling (BFS) and Depth-first Sampling (DFS) to explore neighborhoods \cite{RN7}.

#### Node2vec+

* Implemented as part of **pecanpy** \cite{RN40}.
* Natural extension of node2vec and handles weighted graph more effectively using a noise threshold (average edge weights). Out edges are determined by considering edge weights connecting the potential next state to the previous state. Considers neighbors of the previous state as out edges if the edge weight is below the average \cite{RN41}.

#### Struc2vec

* First constructs a multi-layer weighted graph to generate context for each node. Then, biased random walk is performed on the multilayer graph to learn node sequences \cite{RN20}.
* These sequences are likely to include nodes that are more structurally similar.
* Applied Dynamic Time Warping (DTW) \cite{RN36} on degree sequences to measure the similarity of local structures \cite{RN37}. Nodes with high structural similarity are close to each other in the embedding space \cite{RN3}.

#### Ripple2vec

* In struc2vec, DTW makes distance comparisons less sensitive to signal transformations as shifting, uniform amplitude scaling or uniform time scaling \cite{RN35}. 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 \cite{RN37}.
* Implemented in the framework of struc2vec by adapting ripple distance to define context graphs \cite{RN37}.
* Helps map dis(similar) nodes to (far) near vectors\cite{RN37}.

## Neural network-based

#### Large-scale Information Network Embedding (LINE)

* Has a carefully designed objective function, optimized by an edge-sampling algorithm, that preserves both first-order and second-order proximities \cite{RN11}

#### Structural Deep Network Embedding (SDNE)

* Semi-supervised \cite{RN10}
* Siamese network consisting of Laplacian Eigenmaps (Matrix factorization-based) \cite{RN39}, which generates network representations by factorizing the Laplacian matrix of the adjacency matrix to exploit the first-order proximity for preserving local network structure along with two deep autoencoders to exploit the second-order proximity for preserving global network structure \cite{RN10, 34}.

#### Variational Graph Auto-Encoders (VGAE)

* Utilizes a GCN encoder and an inner product decoder to learn node embeddings \cite{RN3}.
* Can naturally incorporate node features \cite{RN46}.

## Node proximity for signed networks

In signed networks, impacts of edges are distinguished by assigning ‘-1’ to a negative edge, ‘0’ to an uncertain or unseen edge and ‘1’ to a positive edge \cite{RN50}.

#### Neural network signed network embedding (nSNE)

* Generalized from the second-order node proximity for unsigned networks. Introduced a 2nd condition, that is, if two nodes in a signed network are similar, they not only should satisfy the second-order node proximity (i.e., 1st condition) but also have similar sign context to distinguish nodes that have similar neighbors but different sign patterns
* Provides a unified objective function that can preserve both the node and edge pattern of the network
* Embeddings of nodes and mapping functions are learned from the data via multi-layer perceptron with back-propagation algorithm to optimize mapping functions and embeddings

\cite{RN50}

# Section 3: Experiments

The datasets and optimized learned graph embeddings are available at \href{<https://github.com/tengann/IAV_PPI_Graph_Embedding_Review>}.

## Chosen IAV strains

All experiments in this review was conducted based on four Influenza A Virus (IAV) strains of interest, as listed in table \ref {Table 1}.

Table 1: IAV strains of interest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Strain** | **Subtype** | **Taxonomy ID** | **Abbreviation** | **Reason** |
| A/Puerto Rico/8/1934 | H1N1 | 211044 | PR8 | Commonly used in lab experiments |
| A/California/04/2009 | H1N1 | 641501 | CA04 | 2009 Pandemic strain |
| A/California/07/2009 | H1N1 | 641809 | CA07 | 2009 Pandemic strain  (Contains an “unusual” amino acid ‘X’ in its NP segment) |
| A/Aichi/2/1968 | H3N2 | 387139 | Aichi | H3N2 subtypes of IAV have cause seasonal epidemics since 1968 \cite{RN13} |

## Datasets

In this paper, unique datasets were constructed using the human-virus PPI (HVPPI) \cite{RN5}{Yang, 2020 #5} prediction tool {<http://zzdlab.com/hvppi/predict.php>}, that automatically calculates and outputs the interaction probability of a query protein pair. To determine if two proteins interact, three thresholds, equivalent to specificity controls were provided (\ref {Table 2}).

In HVPPI, an unsupervised sequence embedding approach, doc2vec, was applied to represent protein sequences as rich feature vectors of low dimensionality. Then, these vectors were used as inputs to train a random forest (RF) classifier to predict human-virus PPIs. An unbalanced training dataset was built with a positive-to-negative samples ratio of 1:10. Positive samples were downloaded from the Host-Pathogen Interaction Database (HPIDB) \cite{RN17, 16} V3.0. HPIDB covers manually curated host-pathogen interactions and incorporates molecular interactions from other public protein interaction databases. To limit noise and account for sequence similarity of viral proteins in negative samples, dissimilarity-based negative sampling method \cite{RN56} {Eid, 2015 #56} was used in place of completely random pairing.

Table 2: HVPPI interaction probability thresholds

|  |  |
| --- | --- |
| **Threshold** | **Interaction probability (between 0 and 1)** |
| **Positive** | |
| 0.99 | >= 0.375 |
| 0.95 | >= 0.212 and < 0.375 |
| 0.90 | >= 0.143 and < 0.212 |
| **Negative** | < 0.143 |

### Overview of complete network graph

In this paper, the constructed PPI network is sparse. The network contains more than 10k nodes, and approximately 14% edges.

Table 3: Overview of network graph and edge list

|  |  |  |
| --- | --- | --- |
| Data | Complete | Edge list1 |
|
| # nodes | 15, 685  (41 IAV, 15, 644 Human) | |
| # nodes involved in interaction | 12, 438  (41 IAV,  12, 397 Human) | 12, 437  (41 IAV,  12, 396 Human) |
| # edges | 91, 217  (3738 0.99,  26, 044 0.95, 61, 435 0.90) | 48, 8822  (2872 0.99,  15, 782 0.95,  30,228 0.90) |
| # non-interacting pairs | 550, 187 | |
| HVPPI Score (MIN) | 0.000 | 0.001 |
| HVPPI Score (MAX) | 0.99 | 0.99 |
| Degree (Average) | 14.6675 | 7.8607 |
| Degree (Maximum) | 10, 985 | 10, 982 |
| Average degree (IAV) | ~ 2224.8 | ~ 1192.244 |
| Average degree (Human) | ~ 7.358 | ~ 3.9434 |

[caption]

1. This edge list was used as input for all NRL methods.

All query pairs with interaction probability, calculated by HVPPI, above the 0.90 threshold were considered as interacting. The interaction probability also served as edge weights for node2vec+. For nSNE, ‘-1’ was used to denote a non-interacting protein pair. For all other NRL methods, ‘0’ was used instead.

1. To investigate the robustness of various NRL methods on incomplete interaction network, only 53.5% of the interacting edges were used to construct the input edge list**.** Edges with consistent interaction probability scores calculated by HVPPI on two runs were considered.

### Overview of constructed datasets

Table 4: Overview of constructed datasets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset |  | | | Link Prediction  (Unbalanced dataset)4 | |
|  | Train1 | Test 1  (Network Reconstruction)1 | Test 2  (Experimentally Verified) | Train | Test |
| # nodes involved in interaction | 1098  (19 IAV,  1079 Human) | 376  (16 IAV,  360 Human) | 346  (14 IAV.  332 Human) | 6522  (30 IAV,  6492 Human) | 2681  (28 IAV,  2653 Human) |
| # Positive samples (i.e., interacting pairs) | 4014 | 446 | 4462 | 38, 101 | 4234 |
| # Negative samples (i.e., non-interacting pairs) | 4014 | 446 | 4463 | 152, 404 | 38, 106 |
| HVPPI Score  (MIN) | 0.000 | 0.001 | 0.286 | 0.004 | 0.004 |
| HVPPI Score  (MAX) | 0.99 | 0.931 | 0.988 | 0.845 | 0.743 |

[caption]

1 Edges are found in edge list.

2 Positive samples from HPIDB 3.0 \cite{RN17, 16}. These samples were checked against

HVPPI probability scores and identified to also be interacting.

1. Negative samples from negatome 2.0 \cite{RN14} database.
2. All remaining interacting protein pairs not in edge list made up the positive samples in this dataset.

[91, 217 – 48, 882 = 42, 335 (Split into 90% Training + Validation, 10% Test)]

Ratio (Positive: Negative)

Training + Validation 1 (20%): 4 (80%) – Randomly sampled 4 times as many negative samples

Test 1 (10%): 9 (90%) – Randomly sampled 9 times as many negative samples

This section describes the decisions behind constructing the datasets mentioned in table \ref {Table 4}**.**

#### Network reconstruction dataset

A high-quality NRL method should ensure that the learned low-dimensional representation is able to preserve the original network structure \cite{RN10} {Wang, 2016 #10}. Here, prediction was done on existing links in the network, where ground-truth labels of edges are known. For construction of the positive dataset, to reduce training noise, only proteins pairs with interaction probability above the 0.95 threshold were considered. A subset was randomly sampled, where 10% was held out and used solely in the test dataset. The remaining 90% was split into a 9:1 ratio, for training and validation respectively. To construct the negative dataset, same number of non-interacting edges were randomly sampled and the same train-validation-test split was followed.

#### Unbalanced dataset

As many interaction networks are partial due to knowledge gaps in biology, it is necessary that protein-protein interaction (PPI) methods are able to achieve strong performance even when there are missing links in the interaction network \cite{RN6} {Huang, 2020 #6}. Furthermore, in real-world application, PPI networks are said to be small-world networks \cite{RN55}, where, there exists certain protein nodes with a large number of interaction edges. The average degree among IAV protein nodes is approximately 1192. However, there exists seven IAV nodes with degree of over 1000, inclusive of five nodes with degree over 5000. In particular, the NS1 segment of IAV strain PR8 has degree of 10, 985. Meanwhile, the degree distribution of remaining IAV nodes is as follows: 15 nodes have degree between 100 and 1000, 17 nodes have degree between ten and 100 while two nodes have degree of less than ten. On the other hand, with exception of 567 human proteins, most human protein nodes have less than ten PPIs each. This may result in biasing problems as a single protein may appear many times in the positive dataset, causing the classifier to simply predict pairs containing such proteins as interacting \cite{RN43} {Dunham, 2021 #43}.

#### Experimentally verified dataset

This experimentally verified dataset was constructed to increase the reliability of results, particularly on negative samples. Negatome \cite{RN14} {Blohm, 2014 #14} was applied, where human protein pairs unlikely involved in physical interactions were extracted to form the negative test dataset. As interaction probabilities given by HVPPI is not fully accurate, usability of our randomly sampled negative dataset may be limited and questionable. Therefore, false positive rate in PPI discovery has to be evaluated as among them may exist true positives that are presently unidentified \cite{RN43} {Dunham, 2021 #43}. Negatome was constructed via manual curation of literature and studying three-dimensional structures of protein complexes. To guide the manual annotation process, modified version of text mining tool, based on semantic sentence analysis, Excerbt \cite{RN64} {Barnickel, 2009 #64}, was employed over the full corpus of abstracts in PubMed and PubMed Central (PMC) full-text \cite{RN65} {Coordinators, 2016 #65} articles.

## 3.3 Experiments

### 3.3.1. Models

In this paper, two different models were adopted as final classifiers for prediction. Edge embeddings of protein pairs were used as input to the classifier. After training, the classifier outputs a binary class label, where ‘1’ and ‘0’ respectively denotes whether there exists an interaction or not between two queried proteins.

#### Feedforward neural network (FNN)

The softmax FNN classifier was implemented using deep learning framework, Keras V2.9.0 in Python V3.7.4. The model consists of three hidden layers containing 128, 64 and 32 neurons in each layer respectively. The default number of epochs is 200 and learning rate was set to be 0.001 with Adam optimizer \cite{RN54} {Kingma, 2014 #54}. Dropout, an essential trick commonly used in deep learning \cite{RN12} {Liu, 2020 #12} and early stopping were adopted during training to avoid overfitting. Dropout rate is 0.5.

#### Skip-GNN

In PPI networks, interactions between protein nodes are not certainly direct and may involve nodes that are not similar \cite{RN58} {Kovács, 2019 #58}. Vanilla graph neural network (GNN) is inadequate at completely capturing key information that exists further away as it only takes into account direct similarity between nodes in a network \cite{RN57} {Abu-El-Haija, 2019 #57}. Hence, Skip-GNN \cite{RN6} {Huang, 2020 #6} introduced skip similarity, that is, similarity in second-order proximity interactions from second-hop neighbors, into a GNN. In the skip similarity graph, skipped nodes are embedded close together in the latent space. Different NRL methods were used to learn the network representations from the direct similarity graph, otherwise known as original graph. Thereafter, for the final embedding, integration of an iterative fusion scheme allowed the original graph and skip similarity graph to learn from each other. The implementation code was obtained from {https://github.com/kexinhuang12345/SkipGNN}. The setting of hyper-parameters was as follows: epoch size: 15, mini-batch size: 256, dropout rate: 0.5 and learning rate: 1e-3 using Adam optimizer. Hidden size in the first and second layer were set at 64 and 32 respectively.

### 3.3.2. Hyper-parameter settings

The default hyper-parameters used are as follows:

* Node feature vector dimension was set to 128, with the exception of VGAE and nSNE. For

VGAE, this is dependent on hyper-parameter ‘hidden2’ and for nSNE, output is an edge feature vector with dimension dependent on hyper-parameter ‘K’.

* Except LINE and VGAE, remaining methods were all trained for a single epoch. For VGAE, number of training epochs was set to 200 and for line, it is dependent on the ‘epochs’ hyper-parameter.
* For GraRep, k was set to 2 to preserve second-order proximity of the network by default
* Other hyper-parameters for random walk-based methods: number of parallel processes (i.e., workers): 8 and skip-gram window size: 10.
* Other hyper-parameters for neural network-based methods: dropout rate: 0.5 and learning rate: 0.01

#### Hyper-parameter sensitivity

Sensitive hyper-parameters which were mentioned to be important by their authors and in the general guidelines provided by \cite{RN3} {Yue, 2020 #3} were carefully tuned via grid search. Table \ref {Table 5} shows the definitions and selected optimal hyper-parameters for each NRL method. The effect of these hyper-parameters on each method is shown in *Supplementary material.* Hyper-parameters were tuned based on the network reconstruction dataset. Same hyper-parameters were then applied on both unbalanced and experimentally verified datasets.

Table 5: Optimized hyper-parameters (based on network reconstruction dataset)

|  |  |  |  |
| --- | --- | --- | --- |
| **NRL Method** | **Definition** | **Chosen hyper-parameters** | |
| **FNN** | **Skip-GNN** |
| Matrix factorization-based | | |  |
| GraRep | k-steps = number of transition steps (k-order proximity matrix) | k-steps = 2\* | k-steps = 2\* |
| Random walk-based | | |  |
| deepwalk | num\_walks: number of walks per node  walk\_length: length of each walk  p: return parameter1  q: in-out parameter2 | num\_walks = 16, walk\_length = 32 | num\_walks = 128, walk\_length = 8 |
| Node2vec | num\_walks = 8, walk\_length = 32  p = 0.25 , q= 0.5 | num\_walks = 8, walk\_length = 64  p = 0.5 , q= 0.25 |
| Node2vec+ | num\_walks = 8, walk\_length = 32  p = 0.25 , q= 0.5 | num\_walks = 8, walk\_length = 32  p = 0.25 , q= 0.5 |
| Struc2vec | num\_walks = 128, walk\_length = 16 | num\_walks = 64, walk\_length = 64 |
| Ripple2vec | num\_walks = 8, walk\_length = 8 | num\_walks = 8,  walk\_length = 64 |
| Neural network-based | | |  |
| LINE | epochs: number of training epochs | 1st + 2nd order proximity,  epochs = 10 | 1st + 2nd order proximity,  epochs = 25 |
| SDNE | α: balances the weight of 1st and 2nd-order proximities between nodes  β: modulates the reconstruction weight of non-zero elements in the training graph | α = 0,  β = 10 | α= 0.3,  β=10 |
| VGAE | hidden1: number of units in the  hidden layer  hidden2: dimension of latent variables | hidden1 = 256, hidden2 = 128 | hidden1=16, hidden2=8 |
| Node proximity for signed networks | | |  |
| nSNE | K: Edge feature vector dimension  β: tradeoff parameter between the two conditions in node proximity for signed networks | K = 128,  β = 0.005 | N/A |

[caption]

\* default value

1p value <1 encourages returning back to previous node, and value >1 discourages

2 q value <1 encourages walks to go outwards, and value >1 encourage walks within a localized neighbourhood \cite{RN40}

p=1 and q=1, is a special case of node2vec, where sampling approach in deepwalk can be seen \cite{RN7}

3.3.3. Choosing mapping function to compute edges

In this section, the most desirable mapping function for each NRL method was determined according to their performance across datasets. This selection process was only conducted for embeddings used as input to the FNN model. For Skip-GNN, concatenation was used, as suggested by the original publication. Most existing NRL methods were designed to learn only node vectors. However, link prediction calls for computation of representation for pairs of nodes. Thus, mapping functions are formulated to derive edge embeddings from node embeddings.

*Table 6: Element-wise operators for computation of representation for pairs of nodes, where u and v represent the source and target nodes respectively.*

|  |  |
| --- | --- |
| **Operator** | **Definition** |
| Sum |  |
| Average |  |
| Hadamard |  |
| Concatenation |  |

As presented in figure \ref{Figure 1}, hadamard product is highly stable and performed well consistently when applied on most methods, excluding GraRep and VGAE. For GraRep and VGAE, hadamard was the worst performing on the network reconstruction and unbalanced datasets. As shown in figure \ref{Figure 2}, sum, average and concatenation all performed significantly better than hadamard product. For GraRep, although sum performed best on the network reconstruction and unbalanced datasets, concatenation was chosen as it performed significantly better on the experimentally verified dataset. For VGAE, sum was chosen as it yield consistent robust performance across all three datasets.

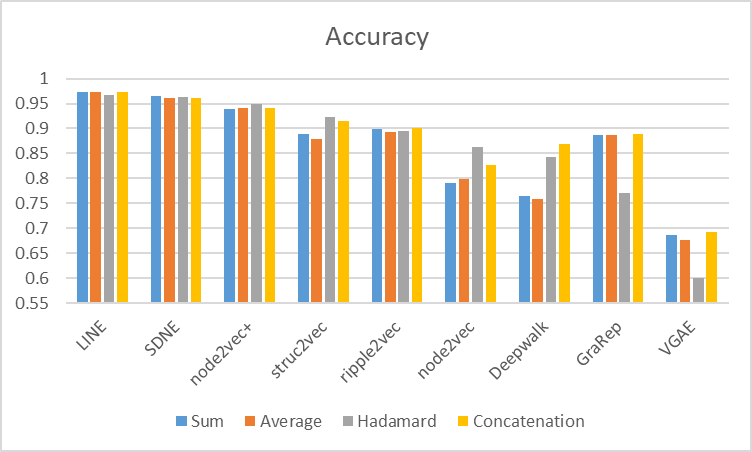
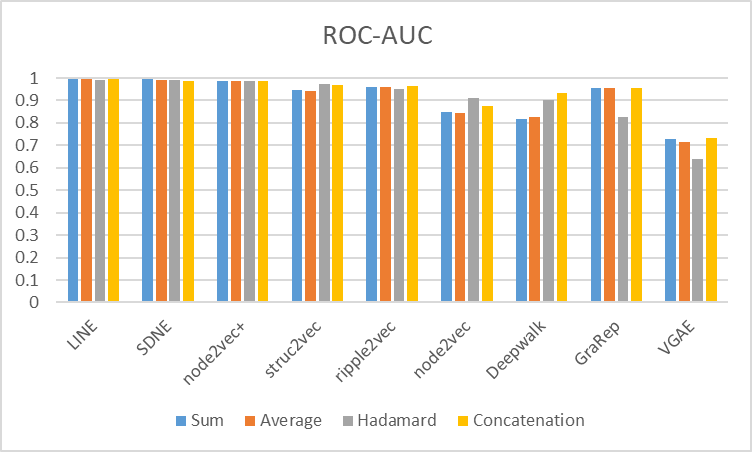


Figure 1(a): Accuracy



*Figure 1(b): ROC-AUC*

[caption]*Figure 1: Comparison of element-wise operators used across NRL methods on the network reconstruction dataset*

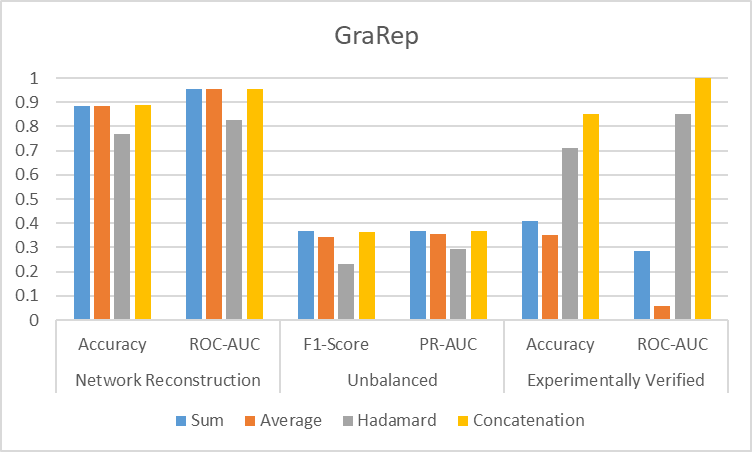


Figure 2(a): GraRep

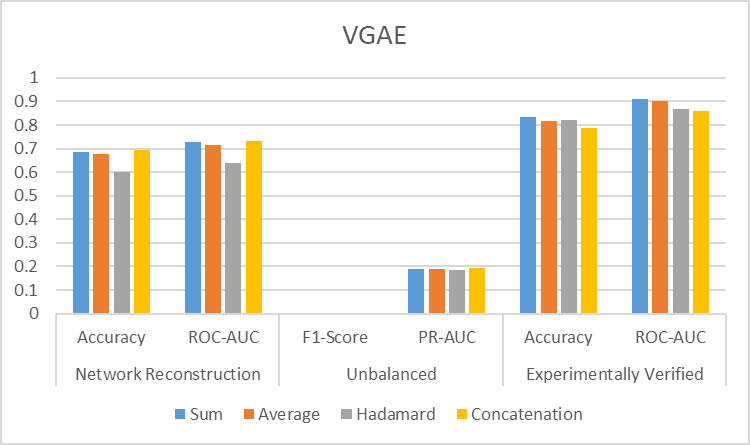


Figure 2(b): VGAE

[caption]*Figure 2:* Choosing of mapping function for GraRep and VGAE

## 3.4 Results

Performance of all NRL methods on the network reconstruction dataset was evaluated on both FNN and Skip-GNN. Performance on the unbalanced dataset was only investigated using FNN. Subsequently, the top four methods that achieved best performance were further explored using the experimentally verified dataset on FNN. For comparison, protein sequence embedding methods were also tested using the experimentally verified dataset, trained on FNN.

The following standard classification metrics were used to evaluate performance of all NRL methods.

1. Accuracy
2. Sensitivity
3. Specificity
4. Precision
5. F1-Score
6. Area under receiver operating characteristic curve (ROC-AUC)
7. Area under precision-recall curve (PR-AUC)

### Network reconstruction dataset

Table 7: FNN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Method | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| Matrix factorization-based | | | | | | | |
| GraRep *(Concatenation)* | 0.8863 | 0.9215 | 0.8511 | 0.8613 | 0.8903 | 0.9559 | 0.9523 |
| Random walk-based | | | | | | | |
| deepwalk | 0.8419 | 0.8502 | 0.8336 | 0.8365 | 0.8432 | 0.9005 | 0.8815 |
| node2vec | 0.8626 | 0.8964 | 0.8287 | 0.8394 | 0.8668 | 0.9124 | 0.8718 |
| node2vec+ | 0.9493 | 0.9713 | 0.9274 | 0.9305 | 0.9504 | 0.9872 | 0.986 |
| struc2vec | 0.9226 | 0.9283 | 0.917 | 0.918 | 0.9231 | 0.9714 | 0.968 |
| ripple2vec | 0.8942 | 0.9067 | 0.8816 | 0.8847 | 0.8955 | 0.9502 | 0.9441 |
| Neural network-based | | | | | | | |
| LINE | 0.967 | 0.9789 | 0.9552 | 0.9562 | 0.9674 | 0.991 | 0.9901 |
| SDNE | 0.9623 | 0.9883 | 0.9363 | 0.9396 | 0.9633 | 0.9912 | 0.9897 |
| VGAE *(Sum)* | 0.6868 | 0.7229 | 0.6507 | 0.675 | 0.6972 | 0.7287 | 0.6782 |
| Node proximity for signed networks | | | | | | | |
| nSNE | **0.9964** | 0.9969 | 0.996 | 0.996 | 0.9964 | 0.9999 | 0.9999 |

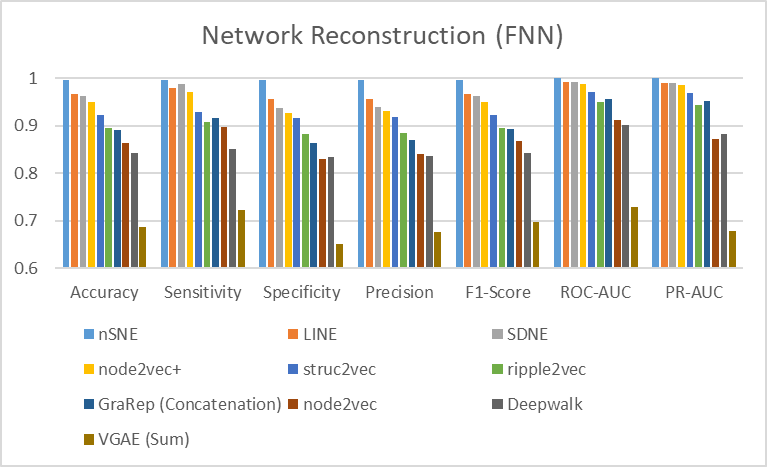


Figure 3: FNN performance comparison on network reconstruction dataset

Table 8: Skip-GNN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Method | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| Matrix factorization-based | | | | | | | |
| GraRep | 0.9154 | 0.8537 | 0.9775 | 0.9745 | 0.9099 | 0.9894 | 0.989 |
| Random walk-based | | | | | | | |
| deepwalk | 0.9393 | 0.9194 | 0.9589 | 0.9575 | 0.9377 | 0.9844 | 0.9838 |
| node2vec | 0.9443 | 0.9156 | 0.9725 | 0.9709 | 0.9421 | 0.9901 | 0.9896 |
| node2vec+ | 0.9461 | 0.9248 | 0.9671 | 0.9649 | 0.9444 | 0.9883 | 0.9874 |
| struc2vec | 0.9487 | 0.9276 | 0.9692 | 0.968 | 0.9471 | 0.9887 | 0.9876 |
| ripple2vec | 0.938 | 0.9106 | 0.9655 | 0.9639 | 0.936 | 0.9854 | 0.9826 |
| Neural network-based | | | | | | | |
| LINE | 0.9362 | 0.903 | 0.9697 | 0.9678 | 0.9341 | 0.9873 | 0.9772 |
| SDNE | 0.9435 | 0.9241 | 0.9627 | 0.962 | 0.9424 | 0.989 | 0.9884 |
| VGAE | 0.9393 | 0.9295 | 0.949 | 0.9481 | 0.9384 | 0.9854 | 0.9848 |

\* nSNE was not tested on Skip-GNN.

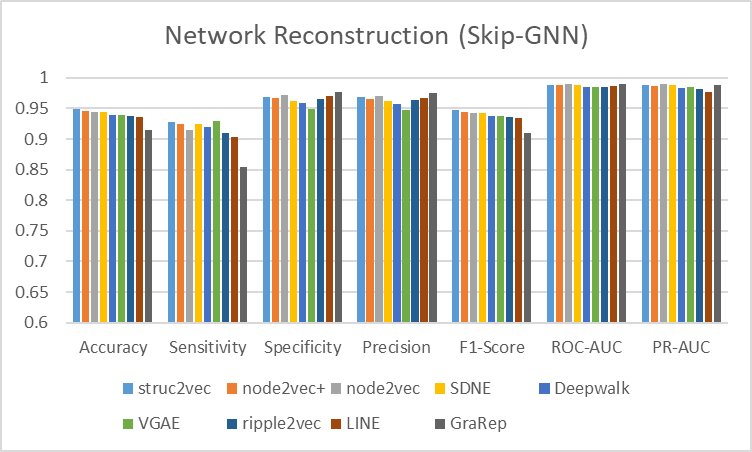


Figure 4: Skip-GNN Performance comparison on network reconstruction dataset

### Unbalanced dataset

Table 9: FNN - Performance of NRL methods on unbalanced dataset

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Method | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| Matrix factorization-based | | | | | | | |
| GraRep *(Concatenation)* | 0.8849 | 0.3377 | 0.9457 | 0.4134 | 0.3681 | 0.8368 | 0.3687 |
| Random walk-based | | | | | | | |
| deepwalk | 0.9076 | 0.1687 | 0.9897 | 0.6614 | 0.2584 | 0.7065 | 0.3203 |
| node2vec | 0.9098 | 0.2722 | 0.9806 | 0.6106 | 0.3762 | 0.7779 | 0.3981 |
| node2vec+ | 0.9187 | 0.6594 | 0.9475 | 0.5829 | 0.6187 | 0.9369 | **0.6609** |
| struc2vec | 0.8975 | 0.2406 | 0.9705 | 0.4764 | 0.3194 | 0.8309 | 0.3723 |
| ripple2vec | 0.8864 | 0.239 | 0.9583 | 0.3893 | 0.296 | 0.7979 | 0.3087 |
| Neural network-based | | | | | | | |
| LINE | 0.9171 | 0.6219 | 0.9499 | 0.5799 | 0.5999 | 0.9323 | **0.646** |
| SDNE | 0.94 | 0.7871 | 0.957 | 0.671 | 0.7241 | 0.9675 | **0.8001** |
| VGAE *(Sum)* | 0.9 | 0 | 1 | 0 | 0 | 0.6881 | 0.1909 |
| Node proximity for signed networks | | | | | | | |
| nSNE | 0.9522 | 0.8504 | 0.9635 | 0.7216 | 0.7806 | 0.9825 | **0.874** |

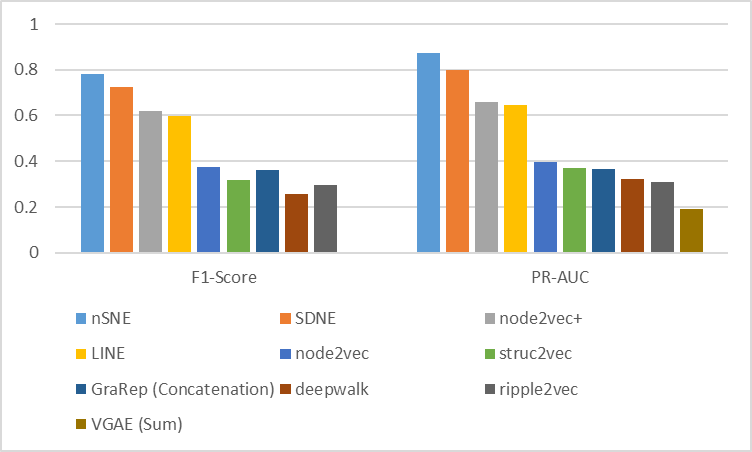


Figure 5: FNN - Performance of NRL methods on unbalanced dataset

### Experimentally verified dataset

Table 10: FNN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| Random walk-based | | | | | | | |
| node2vec+ | 0.5906 | 0.961 | 0.2202 | 0.553 | 0.7017 | 0.802 | 0.7783 |
| Neural network-based | | | | | | | |
| LINE | 0.7848 | 0.9587 | 0.6108 | 0.7129 | 0.8173 | 0.8704 | 0.8067 |
| SDNE | 0.9821 | 0.9843 | 0.9798 | 0.9799 | 0.9821 | 0.9931 | 0.9812 |
| Node proximity for signed networks | | | | | | | |
| nSNE | 0.9388 | 0.9538 | 0.9238 | 0.9346 | 0.9415 | 0.983 | 0.9891 |

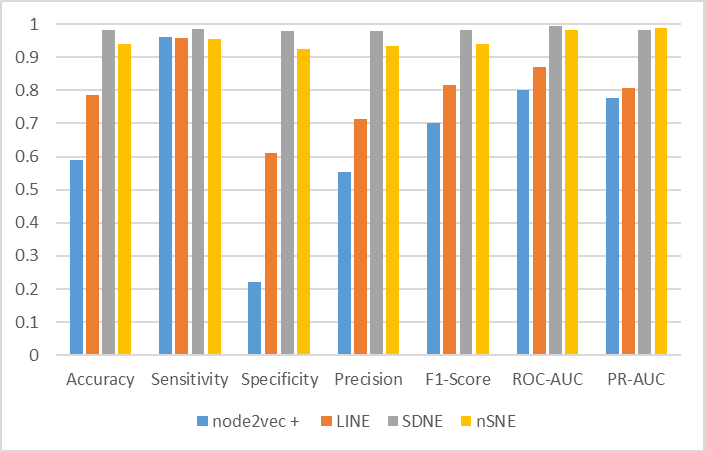


Figure 6: FNN - Experimentally verified

#### Protein sequence embedding

In this section, performance of seven protein sequence embedding methods, namely amino acid composition (AAC) \cite{RN73, 74}, dipeptide composition (DPC) \cite{RN77}, composition, transition and distribution (C/T/D) \cite{RN67, 68}, quasi-sequence-order(QSOrder) \cite{RN70, 80, 79, 71}, amphiphilic pseudo-amino acid composition (APAAC) \cite{RN76, 75}, conjoint triad (CT) \cite{RN69} and normalized Moreau-Broto autocorrelation (NMBroto) \cite{RN78}, were evaluated on the experimentally verified dataset.

Python package, ifeature \cite{RN42} {https://github.com/Superzchen/iFeature} was used to extract features from protein sequences.

Similarly, an optimal mapping function to compute edge from node embeddings was individually selected for all methods.

AAC captures the fraction of 20 natural amino acids within a protein. In addition to the fraction of each amino acid type, DPC also captures information about their local order. In C/T/D, amino acids are divided into three classes, encoded by indices 1, 2 and 3. Each amino acid was then assigned an index, according to which class it belongs. Composition, transition and distribution represents the global percentage for each encoded class in the sequence, transition from a class to another and distribution of each attribute in the sequence respectively. QSOrder was obtained from the Schneider-Wrede physicochemical distance matrix \cite {RN70, 80, 79} and Grantham chemical distance matrix \cite{RN71} between each pair of the 20 amino acids \cite {RN72}. NMBroto lets users select properties from the amino acid (AA) index database \cite{RN42}.

Table 11: Protein sequence embedding

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | **Dim** | Mapping Function | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| AAC | 20 | Sum | 0.7105 | 0.8924 | 0.5287 | 0.6544 | 0.7551 | 0.8137 | 0.7899 |
| DPC | 400 | Average | 0.7103 | 0.9036 | 0.517 | 0.6521 | 0.7574 | 0.8552 | 0.863 |
| C/T/D | 273 | Average | 0.6872 | 0.8897 | 0.4848 | 0.6334 | 0.7399 | 0.7939 | 0.7788 |
| QSOrder | 100 | Average | 0.698 | 0.8906 | 0.5054 | 0.6431 | 0.7468 | 0.7981 | 0.7551 |
| APAAC | 80 | Sum | 0.7058 | 0.8969 | 0.5148 | 0.649 | 0.753 | 0.8123 | 0.7855 |
| CT | 343 | Sum | 0.6892 | **0.9381** | 0.4404 | 0.6266 | 0.7512 | 0.8405 | 0.8412 |
| NMBroto | 240 | Sum | **0.7857** | 0.8816 | **0.6897** | **0.7399** | **0.8045** | **0.8833** | **0.8815** |

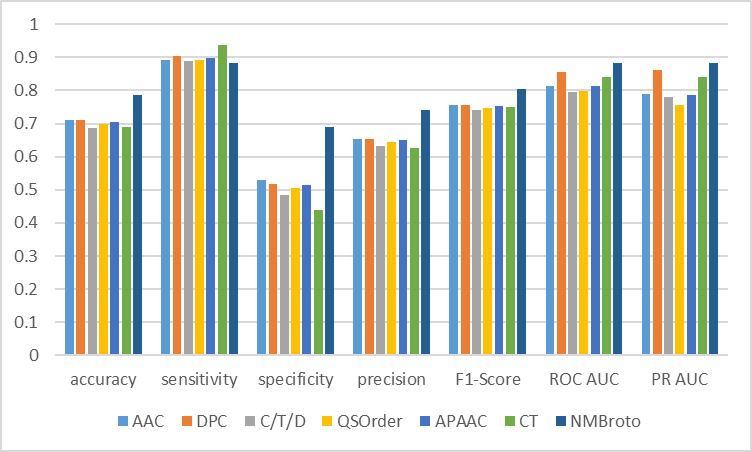


Figure 7: FNN – Experimentally verified, protein sequence embedding

# Section 4: Discussion

Section 4.1 discussed findings based on results obtained in this study. Correspondingly, sections 4.2 and 4.3 further discussed the importance of preserving second-order proximity and edge attributes in the network, based on experiments using the network reconstruction dataset as input to FNN model.

## 4.1 Summary

### Network reconstruction dataset

Generally, based on accuracy scores obtained, except for methods LINE and SDNE, Skip-GNN performed better than FNN.

In Skip-GNN, iterative fusion was implemented for the original and skip similarity graph to interact with each other repeatedly, thereby finding the best dependency structure to form the final embedding. From figure \ref {Figure 4}, although different NRL methods were applied on the original graph, the performance remained almost consistent. Therefore, it was assumed that the final embedding produced is predominantly determined by the skip similarity graph, where skipped nodes from second-hop neighbors are embedded close together in the embedding space. This was in line with an observation made by the author of Skip-GNN, stating that empirically, even simple one-hot position encoding was found to be adequate for Skip-GNN to produce satisfactory results.

In all random walk-based methods that only preserved first-order proximity in the original graph, incorporating skip similarity improved overall performance. However, in LINE and SDNE, where both first and second order proximity were preserved in the original graph, combining skip similarity introduced noise and led to a lower sensitivity score due to higher false negative rate, where protein pairs were erroneously identified as non-edges. This was also observed in GraRep, as presented in table \ref {Table 12}.

Table 12: Skip-GNN GraRep (Network reconstruction dataset)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Proximity (k) | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| 1st-order | 0.9318 | **0.8999** | 0.9639 | 0.962 | 0.9292 | 0.9875 | 0.9867 |
| 2nd-order | 0.9154 | 0.8537 | 0.9775 | 0.9745 | 0.9099 | 0.9894 | 0.989 |

In FNN, the top four best performing methods on the network reconstruction problem in terms of increasing accuracy are: nSNE, LINE, SDNE and node2vec+. As shown in the Uniform Manifold Approximation and Projection (UMAP) \cite{RN59} {McInnes, 2018 #59} plots presented in figure \ref {Figure 6}, these four methods were able to clearly distinguish highly interacting protein pairs (i.e., data points in green) from remaining protein pairs with lower interaction probabilities and the negative pairs. Also, there were minimal overlaps between the three clusters. On the other hand, both struc2vec and ripple2vec failed to tell apart the highly interacting protein pairs despite forming clusters. Node2vec and deepwalk both produced a single cluster with numerous noisy points scattered around.

In FNN, VGAE was not able to achieve favorable results (figure \ref {Figure 3}) as it uses a fixed distribution, Gaussian \cite{RN46} {Kipf, 2016 #46}, to construct latent representation of the network. However, in real world applications, network frequently contain many complex structural properties, for example, first/second-order proximity and community structures \cite{RN60}{Shan, 2020 #60}. Furthermore, integration of Gaussian prior me with inner product decoder caused embeddings to be pushed away from the zero-center \cite{RN46} {Kipf, 2016 #46}. Therefore, in Skip-GNN, when skip similarity was fused with original representation output by VGAE, higher-order proximity was captured in the latent representation which in turn vastly improved performance (figure \ref {Figure 4}).

### Unbalanced dataset

NRL methods were evaluated based on F1-Score and PR-curve as these measures are more suitable and provide reliable information when used on data with uneven class distribution \ref {RN43} {Dunham, 2021 #43}.

Experiments on this dataset revealed that SDNE outperformed LINE. Firstly, as SDNE is a deep model with multiple layers of non-linear functions, it is able to capture highly non-linear network structure. However, LINE is a shallow model, which is not suitable to capture complex and non-linear structure in the underlying network \cite{RN10} {Wang, 2016 #10}. Furthermore, representations output by shallow models are prone to contain missing graph properties information\cite {RN48, 63} {Makarov, 2021 #48} {Galkin, 2020 #63}. Secondly, SDNE is more ideal as it jointly optimize representations for first-order and second-order proximity, while LINE directly concatenate these two representations \cite{RN10} {Wang, 2016 #10}.

For remaining methods, results obtained were similar to that of obtained from experiments on the network reconstruction dataset.

### Experimentally verified dataset

Experiments conducted on this dataset showed that selected NRL methods (\ref {Table 10/Figure 6}) are capable of outperforming protein sequence embedding methods (\ref {Table 11/Figure 7}). As ground truth values of this dataset are highly dependable, low false positive and false negative rates were expected.

Based on sensitivity scores, all four NRL methods achieved scores of above 95%, signifying that false negative rate was low and most true positives were covered. Meanwhile, for protein sequence embedding methods, the maximum sensitivity score obtained was approximately 93.8% by CT, while all remaining methods covered around 90±2% of true positives. For assessment of false positive rate, specificity scores were compared to access coverage of true negatives. In comparison, although Node2vec+ and LINE performed poorly and correctly classified only approximately 20% and 60% of true negatives respectively, SDNE and nSNE were robust and predicted above 90% of true negatives. However, in protein sequence embedding, with the exception of NMBroto, remaining protein sequence embedding methods underperformed, recognizing only approximately half the amount (50%) of true negatives.

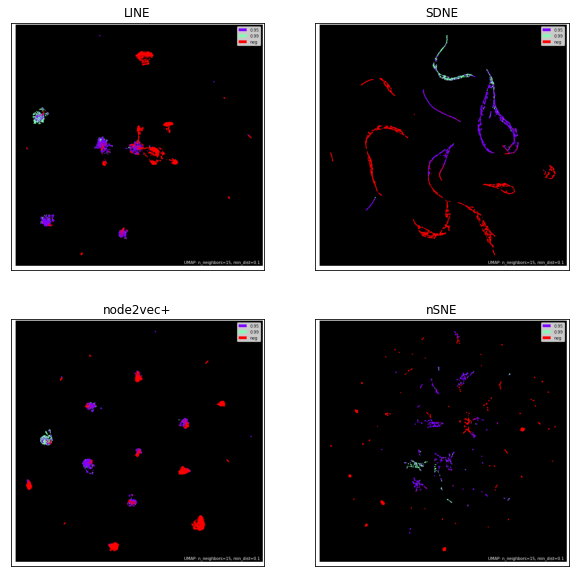
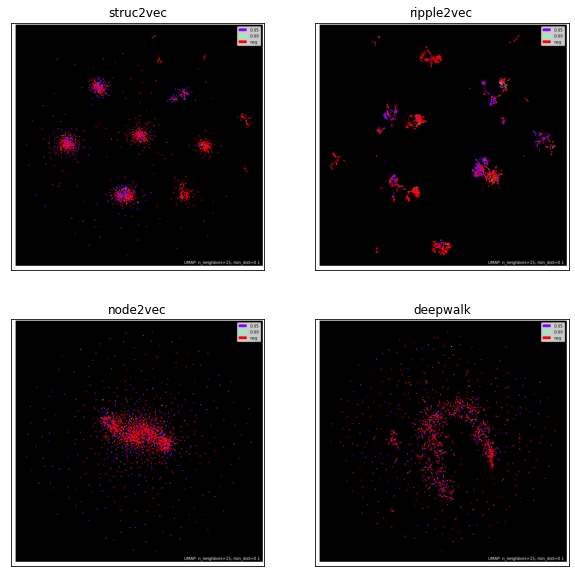


Figure 8: UMAP plot of edges used in the network reconstruction and experimentally verified training set

 [Caption]*: Figure 8: UMAP plot of edges used in the network reconstruction and experimentally verified training set, split into 3 classes. Green, purple and red data points denotes the interacting positive samples within 0.99 threshold, 0.95 threshold and the non-interacting negative samples respectively.*

## 4.2 Second–order proximity

This section highlights the importance of understanding and accounting for similarity between non-connected nodes \cite{RN48} {Makarov, 2021 #48} (i.e., second-order proximity) as nodes with similar neighborhoods would have shared characteristics. First-order proximity and second-order proximity are utilized to capture local and global network structure respectively \cite{RN10} {Wang, 2016 #10}. In the last decade, second-order proximity has been recognized as an exceptionally useful relationship across various interaction networks \cite{RN6} {Huang, 2020 #6}.

In unsigned networks, for neural network-based methods LINE (\ref {Table 13}) and SDNE (\ref {Table 14}), it is evident that accounting for level of proximity directly affects performance. As displayed in figure \ref {Figure 9}, for LINE, when only first-order proximity was employed, all clusters had poor separation. Conversely, when only second-order proximity was employed, distinguishable clusters were formed and accuracy of predictions improved by approximately 15%. Accuracy score further increased when both first and second-order proximities were applied. A similar effect was observed in SDNE, where the best performance was achieved when hyper-parameter, α=0, whereby performance was fully controlled by the second-order proximity. However, as SDNE simultaneously augments first and second-order proximities, it is more robust to sparse networks \cite{RN10}. Thus, it is still able to achieve promising results as α increases and model starts concentrating more on the first-order proximity.

Though, an exception was seen when higher-order proximity was applied to GraRep (\ref {Table 15}). As level of proximity is indirectly associated with community structure \cite{RN48} {Makarov, 2021 #48}, it is a drawback that GraRep it is not clearly community aware. That is, it does not embed communities in a low-dimensional space nor identifies communities in node embeddings \cite{RN66} {Cavallari, 2017 #66}. Additionally, in GraRep, higher-order proximity was extracted from the first-order network structure and only takes a static order of dependency \cite{RN81} {Saebi, 2020 #81}. Therefore, GraRep achieved the best result when hyper-parameter, k=1, when only first-order proximity was preserved instead.

Table 13: LINE: Study of second-order proximity

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| 1st | 0.8038 | 0.7695 | 0.8381 | 0.8264 | 0.7967 | 0.8653 | 0.8564 |
| 2nd | 0.9592 | 0.9713 | 0.9471 | 0.9484 | 0.9597 | 0.9889 | 0.988 |
| 1st + 2nd | 0.967 | 0.9789 | 0.9552 | 0.9562 | 0.9674 | 0.991 | 0.9901 |

[Caption] The number of training epochs was set to 10 for all.

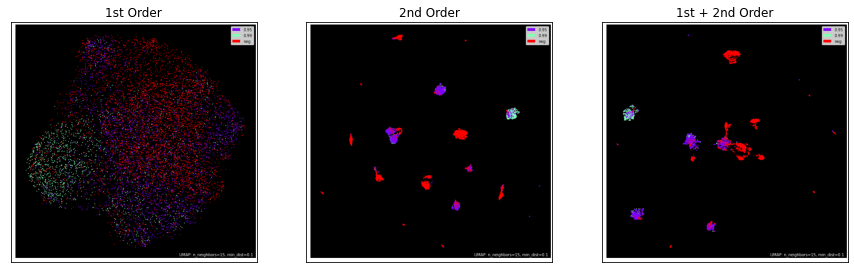


Figure 9: LINE: UMAP – Study of second-order proximity

Table 14: SDNE: Study of second-order proximity

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| α | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| 0 | 0.9619 | 0.991 | 0.9327 | 0.9365 | 0.963 | 0.9909 | 0.9895 |
| 0.1 | 0.9018 | 0.9888 | 0.8148 | 0.8422 | 0.9097 | 0.9561 | 0.9321 |
| 0.2 | 0.9473 | 0.9552 | 0.9395 | 0.9409 | 0.9473 | 0.9809 | 0.9688 |
| 0.3 | 0.9464 | 0.9673 | 0.9256 | 0.9286 | 0.9475 | 0.9855 | 0.9827 |
| 0.4 | 0.902 | 0.9883 | 0.8157 | 0.8428 | 0.9098 | 0.9591 | 0.9437 |

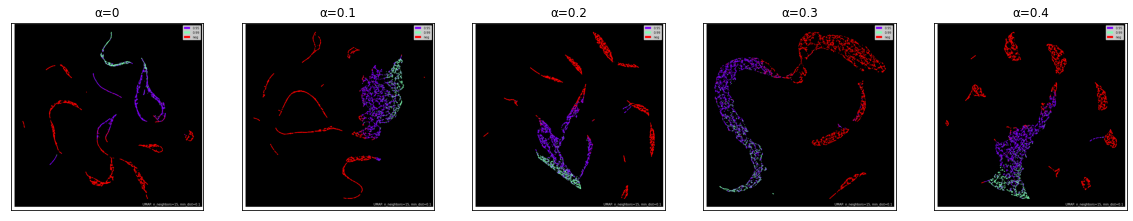


Figure 10: SDNE: UMAP – Study of second-order proximity

Table 15: GraRep: Study of higher-order proximity

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Proximity (k) | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| 1 | **0.9025** | 0.926 | **0.8789** | **0.8844** | **0.9047** | **0.9641** | **0.9615** |
| 2 | 0.8863 | 0.9215 | 0.8511 | 0.8613 | 0.8903 | 0.9559 | 0.9523 |
| 4 | 0.7099 | **0.9825** | 0.4372 | 0.6358 | 0.772 | 0.7378 | 0.6743 |

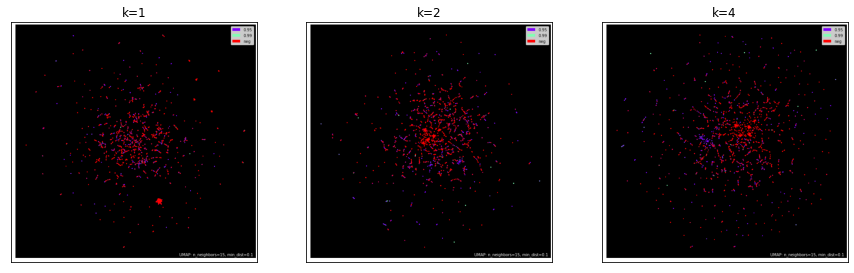


Figure 11: GraRep: UMAP – Study of second-order proximity

## 4.3 Preservation of edge attributes

This section highlights the importance of preserving edge attributes. As transformation of edge information to corresponding nodes are done based on some assumptions, significant edge attributes, for example, edge weights, may be omitted \cite{RN50}.

#### Choosing mapping function

As mapping functions to devise edge from node embeddings are inconsistent and to a great extent, dependent on the data or task, a mapping function has to be specifically chosen through performance evaluation for each candidate function \cite{RN50}. Since nSNE had overcame the problem of inconsistency via direct use of edge vectors, it outperformed all node embedding methods in both network reconstruction and unbalanced dataset and achieved competitive results with SDNE, which was the best node embedding method, on the experimentally verified datset.

In Figures \ref {Figure 13, 14}, UMAP of learned network representations using methods GraRep and VGAE were presented to illustrate an example. Edge embeddings computed by mapping functions sum, average and concatenation are similar and produced comparable results. In contrast, there was approximately 12% and 8% drop in accuracy when hadamard product was utilized for GraRep and VGAE respectively (\ref {Figure 2a, 2b}).

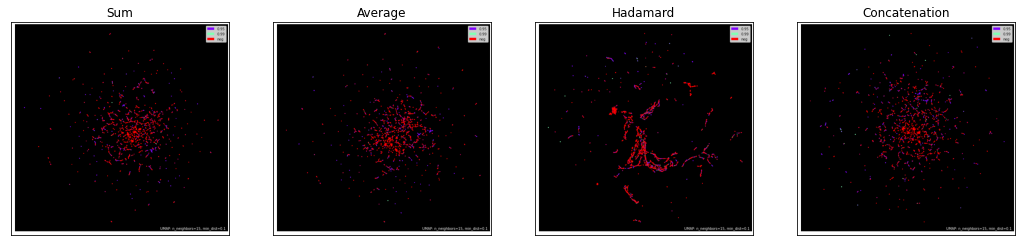


Figure 12: GraRep - UMAP

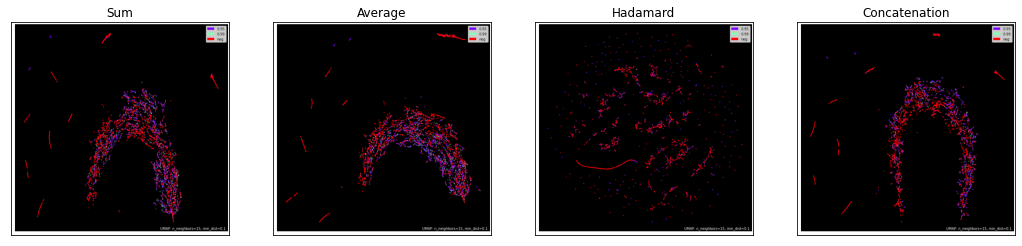


Figure 13: VGAE - UMAP

#### Effect of edge weights

Edge weights are beneficial for graph representational learning. Compared to a simple network of binary edges, more relevant information can be encoded when edge weights are associated \cite{RN84}.

Although LINE and SDNE can be applied to weighted networks, they did not explicitly consider the weights of edges when constructing the adjacency matrix. Furthermore, in LINE, it is challenging to leverage edge weights as problems of high variance and difficulty in finding an appropriate learning rate will arise due to multiplication of edge weights with gradient. In Node2vec, original weights in the graph were not leveraged. Instead, two hyper-parameters, p and q were introduced to search the graph structure with BFS and DFS \cite{RN33}. As BFS and DFS are blind uniform search algorithms \cite {RN31}, constraints on the range of random walks are lacking \cite{RN82}.

Conversely, node2vec+, which took into consideration edge weights when calculating walk biases and was more robust to additive noise in weighted graphs \cite{RN41}, outperformed traditional node2vec in all three datasets (\ref {Figure 14}). This indicated that node2vec+ successfully leveraged on the probability score provided by HVPPI, which served as edge weights. Particularly, in the unbalanced dataset, PR-AUC improved significantly by approximately 25%, which revealed that node2vec+ can be effectively applied to a real-world scenario of PPI prediction, where there may be high imbalance in the natural distribution of class labels.

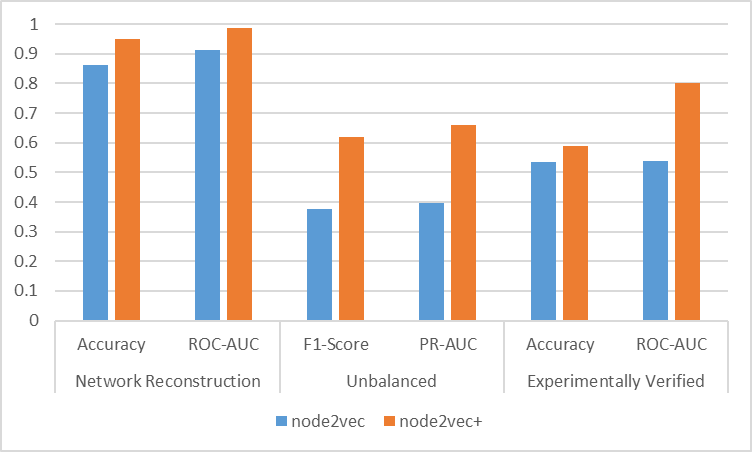


Figure 14: Performance of node2vec VS node2vec+

#### nSNE

As nSNE was employed on signed network, in the adjacency matrix, -1 and 0 were used to represent negative and unobserved edge respectively. Therefore, to differentiate between nodes that have similar neighbors but different sign patterns, nSNE proposed that if two nodes in a signed network are similar, they should not only fulfil the condition of second-order node proximity but also have similar sign context, which was realized when hyper-parameter, 0 < β < 1.

It was determined that satisfying the second condition of similar sign context (i.e., β = 0) was more crucial than satisfying the first condition of second-order node proximity (i.e., β = 1). As reflected in table \ref {Table 16}, there was approximately 13% drop in accuracy when β was 1 instead of 0. This proved that signed adjacency matrix helped strengthen impact of both existing observed interactions and negative impact of non-interacting protein pairs \cite{RN22} {Yang, 2020 #22}.

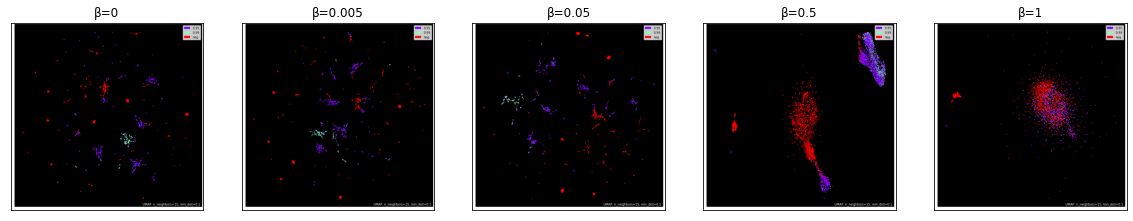


Figure 15: nSNE - UMAP

Table 16: nSNE – Study of hyperparameter β

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| β | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| 0 | 0.9957 | 0.9964 | 0.9951 | 0.9951 | 0.9957 | 0.9999 | 0.9999 |
| 0.005 | 0.9964 | 0.9969 | 0.996 | 0.996 | 0.9964 | 0.9999 | 0.9999 |
| 0.05 | 0.9942 | 0.9955 | 0.9928 | 0.9929 | 0.9942 | 0.9999 | 0.9999 |
| 0.5 | 0.9791 | 0.9865 | 0.9717 | 0.9722 | 0.9793 | 0.9984 | 0.9984 |
| 1 | 0.8652 | 0.8664 | 0.8641 | 0.8646 | 0.8654 | 0.9348 | 0.9244 |

## 4.4 Limitations

Ultimately, although HVPPI is a useful tool, it is still based on predictions. Therefore, it is not fully accurate. Datasets constructed may contain slight deviations.

Unlike learning of protein sequence embedding, where standalone protein sequence can be retrieved using online databases and directly used as input to the feature extraction model, learning of graph embeddings require construction of a network with sufficient number of known edges as input.

# Section 5: Future work

Firstly, for a more objective review, this study can be extended to take into account more IAV strains.

Secondly, instead of straightforward binary output interaction probabilities of protein pairs should be calculated, so that highly positive or negative edges can be determined and more focus and priority can be placed on them for benchwork experiments.

Lastly, to aid with assessment and filtering of false positives and false negatives, there is a need to combine computational methods with experimental techniques for more reliable results and improve interactome coverage \cite{RN51} {Sarkar, 2019 #51}.

# Section 6: Conclusion

Although Skip-GNN generally performed better than FNN, results were comparable with neural network based methods LINE and SDNE performing better. Furthermore, direct application of edge embeddings on FNN (i.e., nSNE) was able to achieve promising results. Both SDNE and nSNE, which are able to overcome problem of sparsity in the network performed best across all three datasets.

Overall, methods that preserve higher-order proximity and edge attributes improves performance. This paper demonstrated that a classifier without complex neural network structures is likewise capable of achieving favourable performance as embeddings already contain sufficient information and are highly representative in the learned low-dimensional vector space, as mentioned by \cite{RN22}{Yang, 2020 #22}.

Additionally, based on results obtained using the experimentally verified dataset, graph embedding methods are able to achieve comparable results or even outperformed protein sequence embedding methods.