Machine learning methods used in conjunction with high-throughput experimental techniques {Sarkar, 2019 #51}

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

## Matrix factorization-based

#### GraRep

## Random walk-based

#### Deepwalk

#### Node2vec

#### Node2vec+

#### Struc2vec

#### Ripple2vec

## Neural network-based

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

#### Structural Deep Network Embedding (SDNE)

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

## Node proximity for signed networks

A definition that is generalized from the second-order node proximity for unsigned networks (e.g. LINE) [1st condition]

Neural network signed network embedding (nSNE) 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 but also have similar sign context.

[Demonstrated by tradeoff parameter Beta]

Provides a unified objective function that can preserve both the node and edge pattern of the network

nSNE: Embeddings of nodes and mapping functions learned via back-propagation algorithm

# Section 3: Experiments

The datasets and learned embedding vectors are available at: <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 \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

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. Human-Human protein

interactions only, involving 482 Human Protein Nodes.

1. 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 **\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. As interaction probabilities given by HVPPI is not fully accurate, usability of our randomly sampled negative dataset may be limited and questionable. Negatome \cite{RN14} {Blohm, 2014 #14} was applied to evaluate false positive rate in PPI discovery as there may exist true positives that are presently unidentified. Human protein pairs that are not likely involved in physical interactions formed the negative test dataset.

Unbalanced and experimentally verified datasets were only used as input data to FNN.

## 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, remaining methods were all trained for a single epoch.
* 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. \ref {Table 5} shows the definitions and selected 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-steps = 2 | k-steps = 1 |
| 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]

1 p 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.

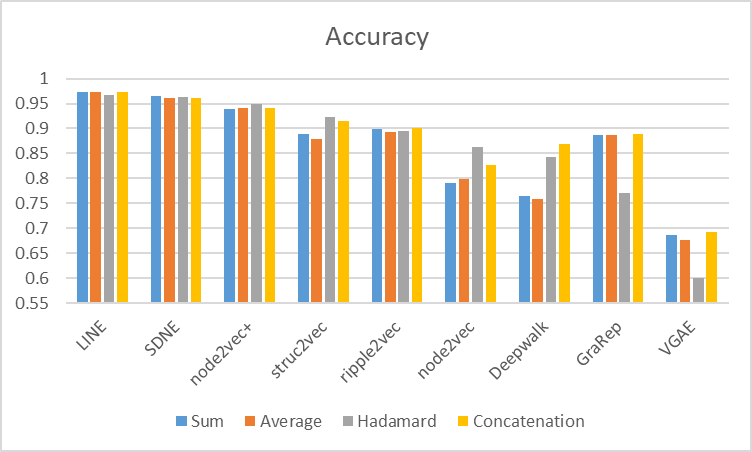
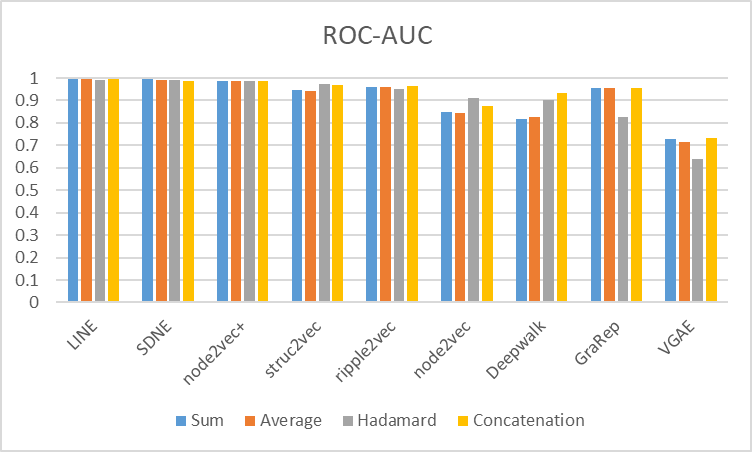


Figure 1(a): Accuracy



*Figure 1(b): ROC-AUC*

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

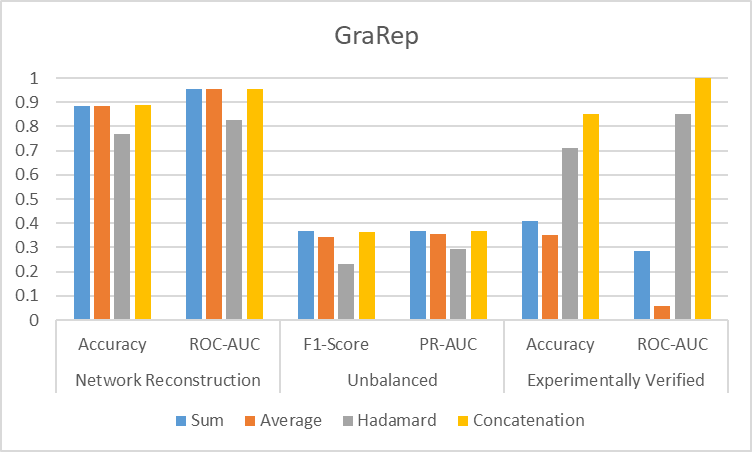


Figure 2(a): GraRep

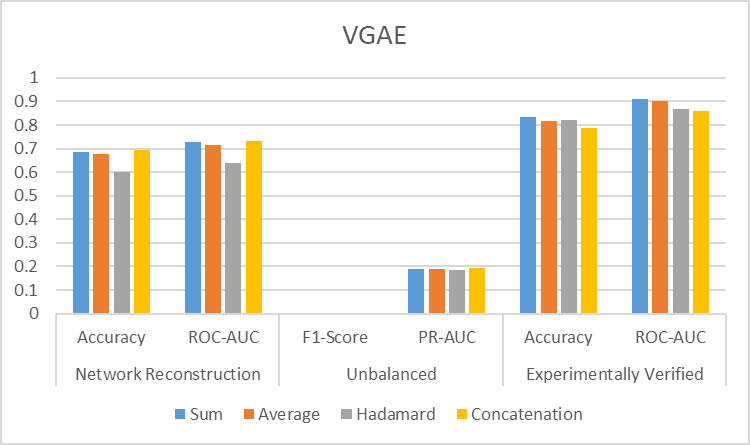


Figure 2(b): VGAE

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

## 3.4 Results

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 |

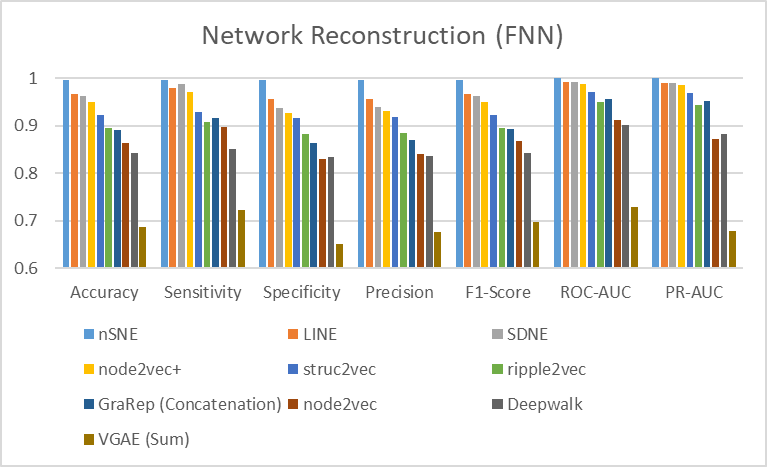


Figure 3: FNN performance comparison on network reconstruction dataset

Table 8: Skip-GNN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| Matrix factorization-based | | | | | | | |
| GraRep | 0.9318 | 0.8999 | 0.9639 | 0.962 | 0.9292 | 0.9875 | 0.9867 |
| 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.

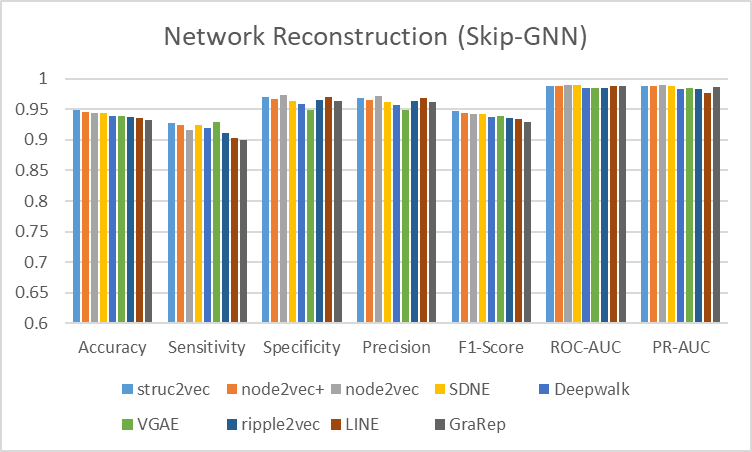


Figure 4: Skip-GNN Performance comparison on network reconstruction dataset

### 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** |

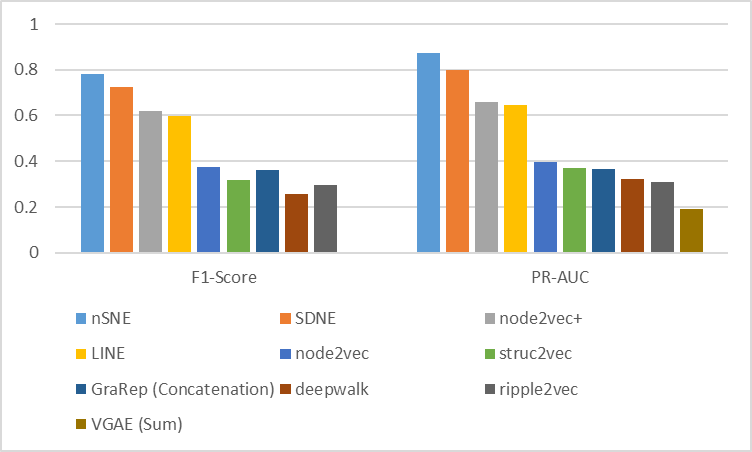
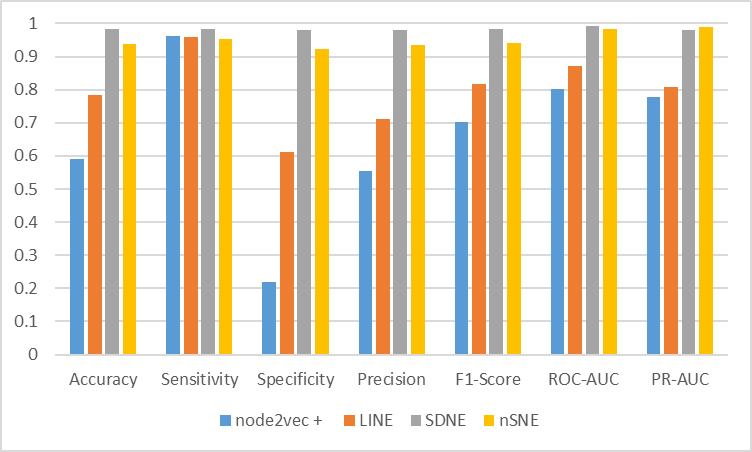


Figure 5: FNN - Performance of NRL methods on unbalanced dataset

### 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 |



#### Comparison with protein sequence embedding

In this section, performance of seven protein sequence embedding methods, namely (AAC), (DPC), (C/T/D), (QSOrder), (APAAC), (CT) and normalized Moreau-Broto autocorrelation (NMBroto), were evaluated on the experimentally verified dataset. AAC…

AAC and DPC 20 amino acid types

NMBroto lets users select properties from the amino acid (AA) index database \cite{RN42} {Chen, 2018 #42}.

Comparison

Table 11: Protein sequence embedding

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | **Dim** | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| AAC | 20 | 0.6742 | 0.8928 | 0.4556 | 0.6212 | 0.7327 | 0.7211 | 0.6404 |
| DPC | 400 | 0.6599 | 0.9018 | 0.4179 | 0.61 | 0.7267 | 0.7477 | 0.672 |
| C/T/D | 273 | 0.5 | 0.6 | 0.4 | 0.3 | 0.4 | 0.5061 | 0.5076 |
| QSOrder | 100 | 0.6733 | 0.8753 | 0.4713 | 0.6249 | 0.7281 | 0.7074 | 0.6189 |
| APAAC | 80 | 0.6778 | 0.8874 | 0.4682 | 0.6253 | 0.7336 | 0.7184 | 0.6338 |
| CT | 343 | 0.6251 | 0.8906 | 0.3596 | 0.5822 | 0.7038 | 0.7363 | 0.6793 |
| NMBroto | 240 | 0.7744 | 0.8529 | **0.696** | 0.7404 | 0.7905 | 0.8233 | 0.7593 |

[Insert plot]

[Provide brief description of protein sequence embedding methods] – 1 to 2 liners

# Section 4: Discussion

## Network reconstruction dataset

Skip-GNN

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 \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.

When results are compared against that of given by FNN, with the exception of GraRep, incorporating skip similarity into all random walk-based methods and VGAE that only preserved first-order proximity in the original graph, improved performance. However, on LINE and SDNE, where both first and second order proximity were preserved in the original graph, combining with skip similarity introduced noise to the final embeddings thus deteriorating performance.

FNN

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.

GraRep \ref {Figure 7}🡪 Preservers higher-proximity order. However, … (reason??)

VGAE \ref {Figure 8} 🡪 worst performance (reason??)

## 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}.

With exception that LINE performance dropped

**[Highlight advantage of SDNE over LINE] 🡪 joint optimization VS concatenation**

Experiments on this dataset revealed that SDNE outperforms LINE.

Present homogeneity score of nSNE and SDNE compared to LINE and node2vec+

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

## Experimentally verified dataset

[Comparison with protein sequence embedding]

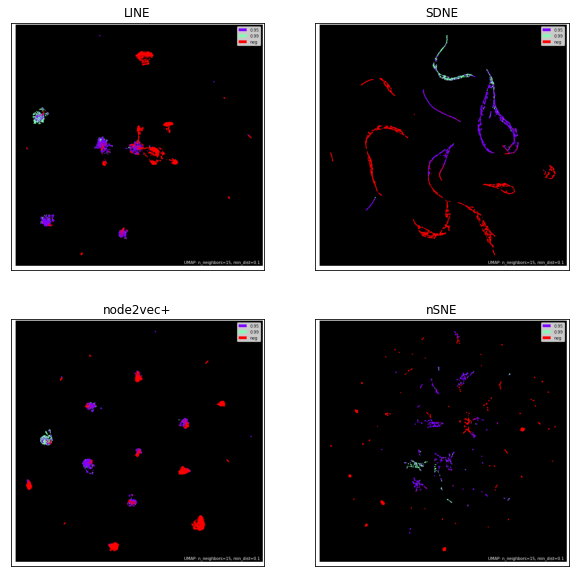
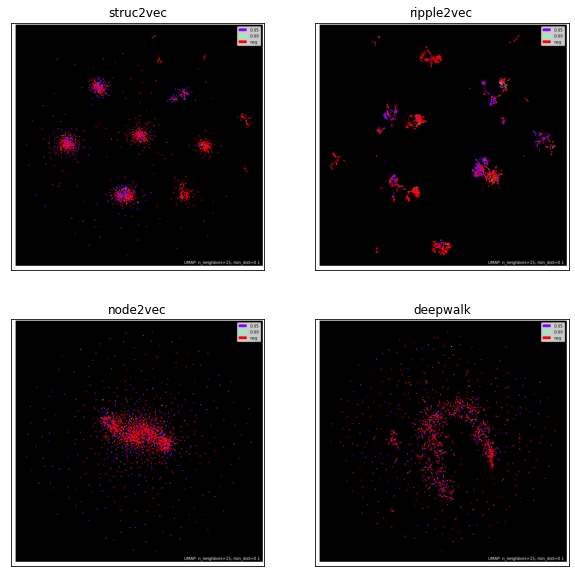


Figure 6: UMAP plot of edges used in the network reconstruction and experimentally verified training set

 [Caption]*: Figure 6: 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.*

### 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. In the last decade, second-order proximity has been recognized as an exceptionally useful relationship across various interaction networks \cite{RN6} {Huang, 2020 #6}.

As presented in \ref {Table 12} LINE method on the network reconstruction dataset.

From the corresponding UMAP plots displayed in \ref {Figure 9},

Using 1st instead of 2nd-order proximity improves accuracy of predictions by approximately 15%

Similarly, for SDNE, the best performance was achieved when hyper-parameter, α=0, whereby performance was fully controlled by the 2nd-order proximity (\ref {Table 13}).

slight difference between second-order and considering both first and second order

Also highlight that nSNE was designed to fulfil the condition of 2nd-order proximity by default

#### LINE

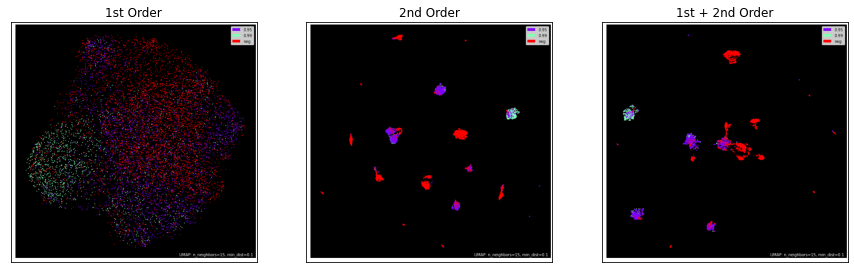


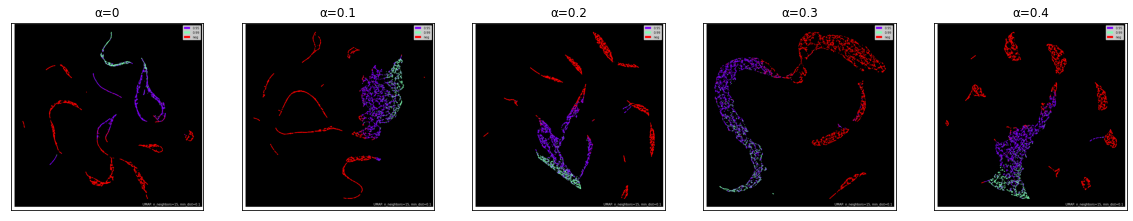
Figure 9: LINE:- UMAP –Network reconstruction dataset

Table 12: LINE:-Results–Network reconstruction dataset

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Accuracy | Sensitivity | Specificity | Precision | F1-Score | ROC-AUC | PR-AUC |
| 1st Order | 0.8038 | 0.7695 | 0.8381 | 0.8264 | 0.7967 | 0.8653 | 0.8564 |
| 2nd Order | 0.9592 | 0.9713 | 0.9471 | 0.9484 | 0.9597 | 0.9889 | 0.988 |
| 1st + 2nd Order | 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.

#### SDNE



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

### Preserving edge attributes

Why is edge attributes important 🡪 related to choosing of mapping function to prevent omission of important edge properties

#### Choosing mapping function - GraRep & VGAE

Sum, average and concatenation – similar clustering evaluation scores

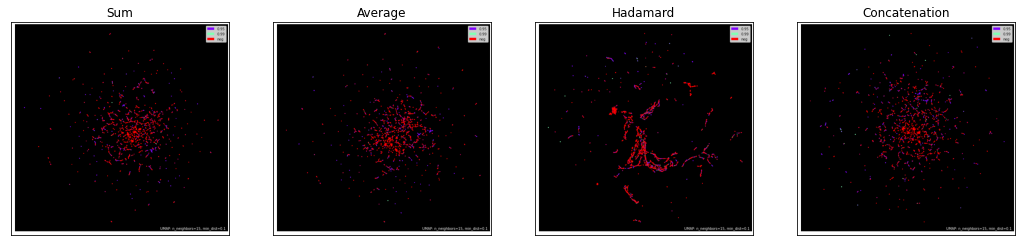


Figure 7: GraRep:- UMAP –Network reconstruction dataset

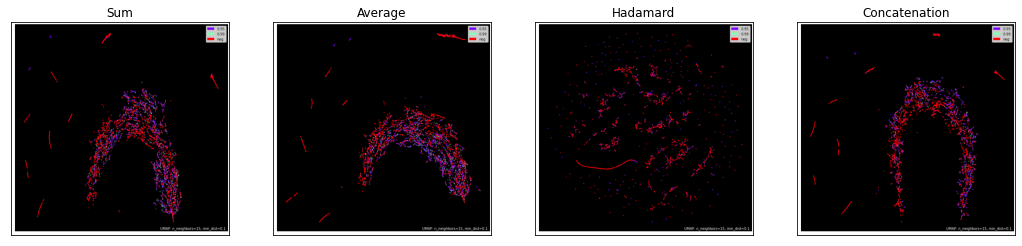


Figure 8: VGAE:- UMAP –Network reconstruction dataset

#### Effect of edge weights

Node2vec VS node2vec+

Weights on edges are useful for graph representational learning \cite{RN33}

How having edges weights instead of binary (1/0) improve performance

(Relate to how node2vec+ handles edge weights)

[Insert results plot – node2vec VS node2vec+ for all 3 datasets]

In the network reconstruction dataset, accuracy improved slightly by approximately 8%

In the unbalanced dataset, PR-AUC improved significantly by almost 30%

LINE and SDNE mentioned that proposed model is able to apply to weighted graph, however, does not explicitly leverage the weights on graph. (construction of the adjacency matrix)

In detail, weight of edge will be multiplied by the gradient which cause a high variance problem and is hard to find a good learning rate {Wu, 2021 #33}.

#### nSNE

Using -1 to represent a negative edge instead of 0

signed VS unsigned network

Satisfying the 2nd condition (If two nodes in a signed network are similar, they should satisfy the condition of having similar sign context) {Song, 2018 #50} 🡪 How satisfying the 2nd condition improves performance

When β = 1, only the first condition for node proximity of signed networks was used (i.e., they only satisfy the second-order node proximity)

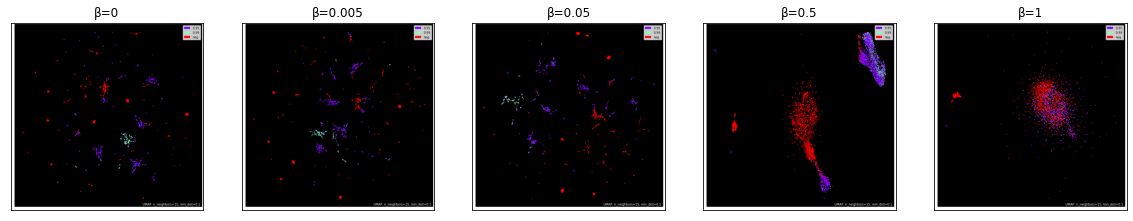


Figure 10: placeholder

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

## Discussion points

* Comparison of Feed-forward neural network VS Skip-GNN (Network Reconstruction Dataset) [\ref{Table 6 and 7}]
* Why I decided to “abandon” Skip-GNN for unbalanced dataset

Method chosen for embedding of original graph does not affect performance of Skip-GNN

Therefore, paper decided to use node2vec

* Choosing mapping function (node 🡪 edge embeddings)

Reason for choosing edges {Song, 2018 #50} 🡪 Problem with converting node to edge embeddings

May omit important edge properties

Performance heavily dependent on dataset or task

(How nSNE outperform all other methods that computes node embeddings)

* Node2vec VS node2vec +
* Weights would help the random walk to focus more on the relevant nodes in the graph
* LINE (1st order proximity VS 2nd and 1st + 2nd order proximity)
* Including reason for SDNE to perform better
* Neural network methods (i.e., LINE, SDNE and nSNE which uses a MLP model) able to perform better on link prediction task, even when dataset is unbalanced (Looking at PR-AUC scores)
* SDNE and nSNE clustering evaluation scores significantly higher/better than all other methods

Better clustering evaluation scores (Homogeneity) = Better performance on a “new” unseen dataset (i.e., experimentally verified)

From **Specificity** scores on the experimentally verified dataset, SDNE and nSNE have lower false positive rate (0 predicted as 1). Even though testing was conducted on human-human interactions instead of iav-human interactions

* Node2vec+ and LINE able to identify the experimentally verified positive samples (high sensitivity score), however not the human-human negative samples (high number of false positives, leading to low specificity and precision scores)
* Explain negative adjusted rand scores for node2vec+ and LINE (random)

## Clustering evaluation

1. Homogeneity, completeness and V-measure {Rosenberg, 2007 #53}

Homogeneity: Each cluster contains only members of a single class

Completeness: All members of a given class are assigned to the same cluster

V-measure: Harmonic mean of homogeneity and completeness

Β is set as default

1. Fowlkes-Mallows score

Geometric mean of the pairwise precision and recall

1. Adjusted rand index

Function that measures the similarity of two assignments

Takes into account that random chance will cause some (data points) to occupy the same clusters

1. Adjusted mutual information score

Measures the agreement of two assignments (split between clusters and split between the ground truth class labels)

Normalized against chance

K-means algorithm ran on the embedded edge vectors to cluster all edges in edge list into 4 classes (positive samples split into 0.99, 0.95, 0.99 thresholds and negative samples).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | Homogeneity | Completeness | V-measure | Fowlkes-Mallows score | Adjusted rand score | | Adjusted Mutual Information score | |
| Matrix factorization-based | | | | | | | | |
| GraRep  (Concatenation) | 0.0378 | 0.0195 | 0.0257 | 0.4748 | -0.03 | | 0.0257 | |
| Random walk-based | | | | | | | | |
| deepwalk | 0.0000 | 0.0000 | 0.0000 | 0.4391 | 0.0002 | | 0.0000 | |
| node2vec | 0.0029 | 0.0014 | 0.0019 | 0.4436 | 0.0091 | | 0.0019 | |
| node2vec+ | 0.0178 | 0.0227 | 0.0199 | 0.6898 | -0.0907 | 0.0199 | |
| struc2vec | 0.0162 | 0.0074 | 0.0102 | 0.4369 | 0.0114 | | 0.0102 | |
| ripple2vec | 0.0154 | 0.0072 | 0.0098 | 0.4372 | -0.0035 | | 0.0098 | |
| Neural network-based | | | | | | | | |
| LINE | 0.0196 | 0.0244 | 0.0218 | 0.6946 | -0.0867 | | 0.0217 | |
| SDNE | 0.3909 | 0.2554 | 0.309 | 0.7649 | 0.4221 | | 0.309 | |
| VGAE (Sum) | 0.0471 | 0.0392 | 0.0428 | 0.6022 | -0.1074 | | 0.0427 | |
| Node proximity for signed networks | | | | | | | | |
| nSNE | 0.6555 | 0.3121 | 0.4229 | 0.6032 | 0.239 | | 0.4229 | |

## Limitations

Ultimately, HVPPI is a prediction tool. Therefore, it is not 100% accurate.

Unable to predict interactions involving new nodes (i.e., without any known links – other IAV strains)

The “complete” graph is required

# Section 5: Future work

* This work can be extended to include other IAV strains.
* Probability instead of binary
* \*\* Need to combine computational method with high-throughput experimental techniques for even more reliable results {Sarkar, 2019 #51}

# Section 6: Conclusion

Graph embedding either achieve comparable or outperforms protein embedding (experimentally verified dataset)

Overall, neural network based methods achieved the best performance

Preserve edge attributes and considering higher order proximity improves performance

FNN was the first and simplest type of artificial neural network devised \cite{RN44}{Schmidhuber, 2015 #44}

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 \cite{RN22} {Yang, 2020 #22}