nleval: A Python Toolkit for Generating Benchmarking Datasets for Machine Learning with Biological Networks

Renming Liu [1] ✉ **and Arjun Krishnan** [2]

[1]Department of Computational Mathematics, Science & Engineering, Michigan State University; [2]Department of Biomedical Informatics, University of Colorado Anschutz Medical Campus

✉ **For correspondence:** arjun.krishnan@cuanschutz.edu

Present address: Room 1501 Engineering Building, 428 S Shaw Ln, East Lansing, MI 48824, United Stats

Code availablity: The source code is released under the MIT License and is freely available at https://github.com/krishnanlab/NetworkLearningEval; the accompanying benchmarking repository to reproduce the experiments is available at https://github.com/krishnanlab/nleval-benchmark

Funding: This work was supported by NIH NIGMS R35 GM128765.

Competing interests: The author declare no competing interests.

Abstract

Recent breakthroughs in graph representation learning (GRL) methods have vastly advanced many real-world applications where graph structures naturally arise, such as drug discoveries, molecular property predictions, and social recommendation systems. The rapid developments of GRL with applications in specific scientific domains are largely enabled by graph learning libraries such as PyTorch Geometric, which provide infrastructures that allow domain scientists to share domain-specific benchmarking datasets, and GRL researchers to adapt and design specialized GRL methods for these particular tasks. Meanwhile, over the past two decades, network biology has demonstrated superior value in harvesting biological insights from network data, such as identifying genes' associated functions, diseases, and traits. Nevertheless, the GRL community faces a significant barrier in working with biological networks because of the tedious and specialized (pre-)processing steps required to set up machine learning (ML)-ready benchmarking datasets. Here, we present `nleval`, a Python package containing reusable modules that enable researchers to effortlessly set up `PyG`-compatible ML-ready datasets using data downloaded from public databases. We expect `nleval` to help network biologists set up custom benchmarking datasets for answering specific biological questions of their interests and help GRL researchers adapt these datasets for designing new specialized GRL architectures.

1 Introduction

Biological systems in living organisms consist of remarkably complex interaction networks where biological entities, such as genes and their products, interact to carry out specific biological functions. Network biology (Barabasi and Oltvai, 2004; Barabási, Gulbahce, and Loscalzo, 2011; Ideker and Sharan, 2008; Vidal, Cusick, and Barabási, 2011), developed over the past two decades, has shown great potential in systematically uncovering genes' functions and their relations to human diseases/traits by leveraging the interconnected nature of biological systems. Accurately identifying the genes associated with a particular biomedical term is a vital step towards understanding the biological mechanisms underlying complex biomedical conditions, whose realization may lead to novel and effective treatment strategies for those conditions (X. Wang, Gulbahce, and Yu, 2011; Lee et al., 2011). Most early works in network biology implement network diffusion-type methods based on the principle known as Guilt-by-Association (Oliver, 2000), which assumes that genes

interacting with each other are likely to participate in the same biological functions or pathways. More recently, we have seen an emergence of more powerful and advanced graph machine learning techniques such as graph embeddings (Goyal and Ferrara, 2018; Cai, Zheng, and Chang, 2018) and graph neural networks (Z. Wu et al., 2020; J. Zhou et al., 2020), which have brought network analysis to a new level. Comprehensive reviews are presented by Ata et al. (2021), Yue et al. (2020), and Nelson et al. (2019) on graph embedding techniques, as well as X.-M. Zhang et al. (2021) and Muzio, O'Bray, and Borgwardt (2021) on graph neural networks, with biomedical applications such as disease gene predictions. We also refer readers to Camacho et al. (2018), Zitnik, Nguyen, et al. (2019), and McGillivray et al. (2018) for reviews on the concepts in network biology.

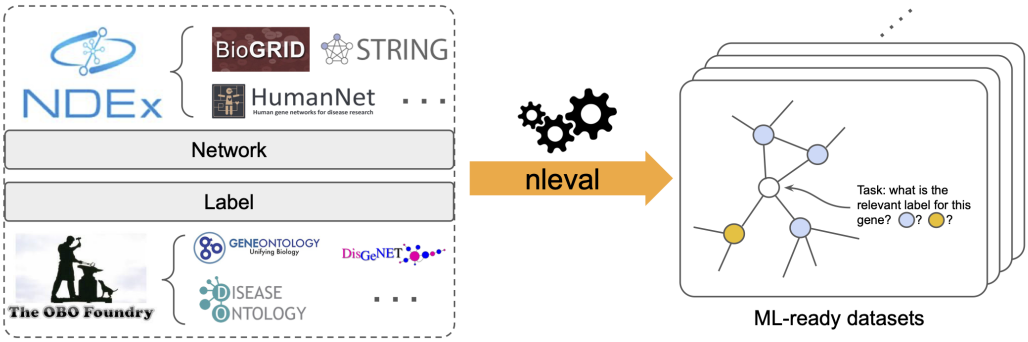
A key to continually developing methods and specialized graph neural network architectures that enhance network biology further is the availability of standardized datasets that allow reliable and reproducible benchmarks (Shchur et al., 2018; Hu et al., 2020). However, setting up machine learning (ML)-ready datasets for network biology is incredibly tedious. Some necessary steps include converting gene IDs, setting up labeled data from annotated ontologies, filtering labeled data based on network gene availability, and constructing realistic data splits mimicking real-world scenarios. As a result, despite the tremendous amount of publicly available data for biological networks (Huang et al., 2018; Ji et al., 2012) and annotations (Smith et al., 2007), the only widely available ML-ready datasets for network biology are `PPI` dataset from OhmNet (Zitnik and Leskovec, 2017) and `PPA` from the Open Graph Benchmark (Hu et al., 2020).

Here, we address these issues by presenting `nleval` (biological **n**etwork **l**earning **eval**uation), a Python package providing unified data (pre-)processing tools to set up ML-ready network biology datasets with standardized data splitting strategies. Our goal is to build a reproducible and transparent data processing pipeline, where each processing step starting from the raw data is open to the public, contrasting to popular benchmarking graph databases such as OGB (Hu et al., 2020) and TUDatasets (C. Morris et al., 2020), where the exact processing pipelines are unknown to the users. At its current state, `nleval` comes with seven genome-scale human gene interaction networks and four collections of gene classification tasks, which can be combined into 28 datasets to benchmark different graph ML methods' capability in network biology. All data are downloaded from various publicly available sources, ensuring that the network and annotation data are most up-to-date. Furthermore, we provide archived data versions to enable exact reproducibility of the datasets.

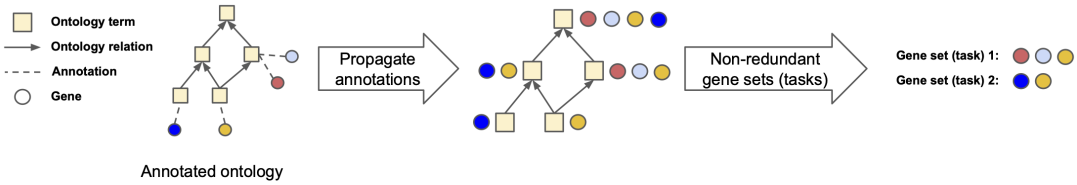
2 Related work

Several existing Python packages share similar goals with `nleval` regarding setting up biological network datasets or providing analysis based on biological networks, whose nodes represent genes or their products, such as protein, and edges represent functional relationships between nodes. `PyGNA` (Fanfani, Cassano, and Stracquadanio, 2020) provides rich functionalities to analyze or visualize one or multiple gene sets using biological networks. `PyGenePlexus` (Mancuso, Liu, and Krishnan, 2022) and `drug2ways` (Rivas-Barragan et al., 2020) provide network-based gene/drug prediction functionalities. `OGB` (Hu et al., 2020) contains a collection of well-set-up graph benchmarking datasets, including a PPI dataset similar to the STRING-GOBP dataset in `nleval` (Section 4). Nevertheless, all four packages above have a limited number of, if any, network data. Since `nleval` implements an interface with `NDEx` to acquire biological network data (Section 3.1), it is much more extensible. `Graphein` (Jamasb, Lió, and Blundell, 2020) focuses on protein and RNA structure analysis using GRL techniques. While it also supports downloading protein interaction data from BioGRID (Stark et al., 2006) or STRING (Szklarczyk et al., 2019), the functionality of setting up network-based gene classification datasets is lacking[1]. Finally, `BioKEEN` (Ali, Hoyt, et al., 2019) (now part of `PyKEEN` (Ali, Berrendorf, et al., 2021)) provides extensive (biological) knowledge graph (KG) datasets and KG embedding models. Although KG tasks (Q. Wang et al., 2017) do not align with

[1] https://github.com/a-r-j/graphein/releases/tag/v.1.5.1



(a) `nleval` downloads network and label data from public data sources, then process and combine them into ML-ready benchmarking graph datasets.



(b) Generating label data from annotated ontology through annotation propagation and non-redundant gene set extraction.

Figure 1. Overview of data processing provided by `nleval`

our settings precisely, there are ways to formulate gene classification tasks into KG completion and vice versa (Ata et al., 2021; Yue et al., 2020). Hence, we plan to implement an interface with `PyKEEN` to further expand our datasets in the near future.

3 Systems description

Making the process of setting up ML-ready network biology benchmarking datasets from publicly available data as effortlessly as possible is the core mission of `nleval`. We implement and package a suite of graph (`nleval.graph`) and label (`nleval.label`) processing functionalities and couple them with the high-level data object `nleval.data` to provide a simple interface for users to download and process biological networks and label information. For example, calling `nleval.data.STRING("datasets")` and `nleval.data.DisGeNET("datasets")` will download, process, and save the STRING network and DisGeNET label data under the `datasets/` directory, which can then be loaded directly next time the functions are called. Users can then compose a dataset object using the network and the label objects, along with the split (Section 3.2.3), which can then be used by a model trainer to train and evaluate a particular graph learning method. Alternatively, the composed dataset object can be transformed into a standard PyTorch Geometric (PyG) (Fey and Lenssen, 2019) data object that is understood by the PyG library.

3.1 Network

Downloading Currently, tens of genome-scale human gene interaction network databases are publicly available, each constructed and calibrated with different strategies and sources of interaction evidence (Huang et al., 2018). Unlike in many other domains, such as chemoinformatics, where there are only a few ways to construct the graph (e.g., molecules), gene interaction networks can be defined and created in a wide range of manners, all of which capture different aspects of the functional relationships between genes. Some broad gene interaction mining strategies include experimentally captured physical protein interactions (Stark et al., 2006), gene co-expression (Johnson and Krishnan, 2022), genetic interactions (Dixon et al., 2009), and text-mined gene interactions (Szklarczyk et al., 2019). We refer readers to Huang et al. (2018) for an in-depth review of 21 popular gene interaction network databases. To overcome the hurdle of loading networks in differ-

ent formats provided by different databases, we download network data from the Network Data Exchange (NDEx) (Pratt et al., 2015) biological network repository Figure 1a, which returns the networks in a consistent `CX` stream format. The `CX` stream is subsequently loaded into a `nleval.graph` object for further processing.

Gene ID conversion In gene interaction networks, each node represents a gene, or its gene product, such as protein. There have been several standards in mapping gene identifiers, and different gene interaction networks might not use the same gene ID. For example, the STRING database (Szklarczyk et al., 2019) uses Ensembl gene ID (Hubbard et al., 2005), PCNet (Huang et al., 2018) uses HGNC (Gray et al., 2015), and BioGRID (Stark et al., 2006) uses Entrez gene ID (Hubbard et al., 2005). Here, we use the NCBI Entrez gene ID due to its advantages, such as supporting more species other than Human and being less ambiguous (D. Himmelstein, C. Greene, and Pico, 2015). To convert other types of gene IDs into the Entrez gene, we use the MyGene query service (C. Wu, Mark, and Su, 2014), which provides the most up-to-date gene ID mapping across tens of gene ID types. We remove any gene where more than one gene ID is mapped to the same Entrez gene, which indicates ambiguity in annotating the gene identifier currently. The gene interaction network after gene ID conversion will contain equal or less number of genes, all of which are one-to-one mapped from the original gene IDs to the corresponding Entrez genes.

Connected component For various reasons, the gene interaction network may not be connected. Particularly, a small fraction (typically within 1%) of genes may be disconnected from the largest connected components (Table S5). The disconnectedness of the network is typically due to missing information about gene interactions from the measurement; the more information a network uses to define the interactions, the denser and less likely there will be disconnected genes (e.g., STRING). Thus, we extract the largest connected component of the gene interaction network by default as it is more natural to have a single component for the transductive node classification tasks we consider. However, a user can also choose to take the full network without extracting the largest connected component by specifying the `largest_comp` option to `False` when initializing the data object.

3.2 Label

Many biological annotations are organized into hierarchies of abstract concepts, known as ontologies (J. B. Bard and Rhee, 2004). Each ontology term (node in the ontology graph) is associated with a set of genes provided by current curated knowledge. By the hierarchical nature of the ontologies, if a gene is associated with an ontology term, then it must also be associated with any more general ontology terms. Thus, we first propagate gene annotations over the ontology, resulting in highly redundant gene set collection. Then, we pick a non-redundant subset of the gene sets with other filtering criteria to set up the final labeled data in the form of $Y \in \{0,1\}^{N \times p}$ with N number of genes and p number of labels (Table 1b).

3.2.1 Annotated ontology

An ontology is a directed acyclic graph $H = (V_H, E_H)$, where $v \in V_H$ is an ontology term, and a directed edge $(u, v) \in E_H$ indicates v is a parent of u, that is, v is more abstract or general concept containing u. Consider B as the set of all genes we are interested in, and $\mathcal{P}(B)$ be the power set of B. Given a gene annotation data, represented as a set of pairs $\mathcal{A} = \{(b, v)_i\}$, where $b \in B$ and $v \in V_H$ are gene and ontology term, we represent the *raw annotation* $J_0 : V_H \to \mathcal{P}(B)$ as a map from an ontology term to a set of genes, so that $J_0(v) = \{b : (b, v) \in \mathcal{A}\}$. We propagate the *raw annotation* J_0 over the ontology H into a *propagated annotation* J, where $J(v) = \cup_{v' : \exists \text{path from} v' \text{ to } v} J_0(v')$ We refer to the images of J_0 and J as the *raw gene set collection* and the *propagated gene set collection*, respectively (first half of Figure 1b).

Downloading All the ontology data will be downloaded from the OBO Foundry (Smith et al., 2007) (Figure 1a), which actively maintains tens of large-scale biological ontologies. In the first release, we focus on two different annotations, Gene Ontology (GO) (Ashburner et al., 2000) and DisGeNET (Piñero et al., 2016), and only consider the `is_a` and `part_of` relations when constructing the ontology graph. We naturally split GO into three different sub-collections, namely biological process (GOBP), cellular component (GOCC), and molecular function (GOMF), resulting in a total number of four different labeled data collections.

3.2.2 Filtering

The propagated annotations can not be directly used as the label data for evaluation for many reasons, chief among them being that they are now highly redundant. We adapt the non-redundant gene set selection scheme used in Liu, Mancuso, et al. (2020) to extract a subset of non-redundant gene sets. In brief, we construct a graph of gene sets based on the redundancy between two gene sets, and recursively extract the most representative gene set in each connected component (second half of Figure 1b). In addition, we provide several other filtering methods such as filtering gene sets based on their sizes, and filtering genes based on existence in the gene interaction network, to help further clean up the gene set collection to be used for final evaluation.

3.2.3 Data splitting

A rigorous data splitting should closely mimic real-world scenarios to provide accurate and unbiased estimations of the models' performance. One of the most stringent solutions is temporal holdout, where the training input and label data are obtained before a specific time point, and the testing data is only available after that (Dessimoz, Škunca, and Thomas, 2013; Liu, Mancuso, et al., 2020). However, this temporal setting is often too strict and leads to an insufficient number of tasks to evaluate (Liu, Mancuso, et al., 2020). Thus, we consider a closely related but less strict strategy called *study-bias holdout* by default.

6/2/2 study-bias holdout We use the top 60% of the most studied annotated genes[2] according to the number of PubMed publications associated with them[3], 20% of the least studied genes for testing, and the rest of the 20% genes for validation. Notice that by splitting up genes this way, some of the tasks may get very few positive examples in one of the splits. Hence, we remove any task whose minimum number of positive examples across splits falls below a threshold value (set to ten by default). For completeness, we also provide functionalities to generate random splits.

3.3 Dataset construction

The network (Section 3.1) and label (Section 3.2) modules provide flexible solutions for processing and setting up datasets. Meanwhile, we provide a default dataset constructor that utilizes the above modules with reasonable settings to set up a dataset given particular selections of network and label. The default dataset construction workflow is the following.

1. First, select the network and label (task collection) of interest.
2. Remove genes in the task collection that are not present in the gene interaction network.
3. Remove tasks whose number of positive examples falls below 50 after the gene filtering above.
4. Setup 6/2/2 study-bias holdout (Section 3.2.3).

Listing 1 shows an example of constructing a default dataset Humanet-DisGeNET with a single line of code. The training and evaluation of a simple GCN model is accordingly done using a simple GNN trainer class with only three extra lines of code.

[2]An annotated gene here means that the gene is a positive example in at least one task.
[3]https://www.ncbi.nlm.nih.gov/

```python
from nleval.util.dataset_constructors import default_constructor
from nleval.metric import auroc
from nleval.model_trainer.gnn import SimpleGNNTrainer
from pytorch_geometric.nn import GCN

# Download, process, and set up a dataset with study-bias split
dataset = default_constructor("datasets", version="latest",
    graph_name="HumanNet", label_name="DisGeNET")

# Set up a GCN model and a simple GNN trainer
gcn_mdl = GCN(in_channels=1, ..., out_channels=dataset.y.shape[1])
trainer = SimpleGNNTrainer(metrics={"auroc": auroc},
    metric_best="auroc")

# Train the model and obtain the model evaluation results
results = trainer.train(gcn_mdl, dataset)
```

Listing 1. Example code for using `nleval` to set up a default dataset HumanNet-DisGeNET, and then train and evaluate a simple GCN model on the dataset. In this case, where the version option is specified to "latest", the raw network and label data will be downloaded directly from the latest source and processed accordingly.

3.4 Community standards

We have made several efforts to ensure community-wide contributions, which is the key to a reusable and maintainable code base. First, we release the code on GitHub https://github.com/krishnanlab/NetworkLearningEval under the permissive MIT license. Second, we use Sphinx to build the documentation of the code base and host it on ReadTheDocs.org https://networklearningeval.readthedocs.io. Several quick-start examples using the package are also provided on the GitHub README page. Third, code quality is ensured via various testing and code-linting automations using tox, pytest, pre-commit hooks, and GitHub actions. Finally, we provide contribution guidelines and a code of conduct on the GitHub page.

4 Experiment

We demonstrate the utility of `nleval` for a network biology practitioner by showing the ease of performing a benchmarking experiment using built-in datasets that can show the need for specialized GNN architectures for network biology tasks. In this experiment, we combine three gene interaction networks (BioGRID (Stark et al., 2006), HumanNet (Hwang et al., 2019), STRING (Szklarczyk et al., 2019)) and two task collections (DisGeNET (Piñero et al., 2016), GOBP (Ashburner et al., 2000)) to form six different datasets, and split the data based on 6/2/2 study-bias holdout (Section 3.2.3). We restrict the genes in the prediction task collections to the set of genes common to all gene interaction networks so that the processed tasks are consistent across networks. The statistics about networks and task collections are shown in Table 1.

Network	# nodes	# edges	Density
BioGRID	18,951	1,103,298	0.0030
HumanNet	17,211	847,104	0.0029
STRING	17,942	10,951,202	0.0340

(a) Network statistics

Label	# tasks	# positives
DisGeNET	123	221.3 ± 124.6
GOBP	203	123.7 ± 33.6

(b) Label set collection statistics

Table 1. Dataset statistics. The network (edge) density is computed as the ratio of existing edges over all possible (undirected) edges: (# edges) / (# nodes × (# nodes - 1)).

4.1 Setup

We first consider five baseline methods, four of which are based on traditional ML classifiers (logistic regression and support vector machines) that use graph-derived features (adjacency matrix (Liu, Mancuso, et al., 2020) or *node2vec* embeddings (Grover and Leskovec, 2016)), and a label propagation method based on random walk with restart (Köhler et al., 2008). We use the default 128 embedding dimensions for *node2vec* and 0.9 restart probability for label propagation. We refer the readers to Appendix A.1 for more detail information about the models and their hyper-parameter selections.

In addition, we consider four popular GNN models: GCN (Kipf and Welling, 2016), GIN (Xu, Hu, et al., 2019), GAT (Veličković et al., 2017), and GraphSAGE (Hamilton, Ying, and Leskovec, 2017). All models are set up using `PyG` 2.0.4[4] `torch_geometric.nn` with five graph convolution layers of 128 hidden features, one pre-preprocessing, and one post-processing (prediction head) layer. We use 32 dimensional one-hot encoded log-uniform node degrees as the input node features (Appendix A.1.2). All models are trained using the standard Adam optimizer (Kingma and Ba, 2015) with a fixed learning rate of 0.001 for a maximum of 20,000 epochs. The optimal epoch is selected based on the validation performance.

The optimal epoch is selected based on the validation performance measured by the log2 fold change of the **a**verage **p**recision score **o**ver the **p**rior (APOP), which is more suitable than commonly-used metrics like the **a**rea **u**nder the **r**eceiver **o**perating **c**haracteristic (AUROC) for gene classification tasks that are highly imbalanced (Saito and Rehmsmeier, 2015; Sofaer, Hoeting, and Jarnevich, 2019) (more information and the motivations about APOP can be found in Appendix A.2). Thus, we base our discussion on APOP throughout the paper, and recommend using this metric for any future benchmarking studies on our datasets. For completeness, we also present performance measured by AUROC.

4.2 Results

4.2.1 Traditional ML methods perform comparably to GNNs overall

Table 2 shows the overall performance for each model on the HumanNet-DisGeNET and HumanNet-GOBP datasets, averaged over five repeated runs. Traditional ML methods using graph-derived features achieve comparable or even better overall performance than GNNs when using HumanNet as the input network. For example, using the rows of the adjacency matrix as features for logistic regression predicts disease genes (DisGeNET) the best. Similar results are obtained when using the other two networks (Table S1,S2). The superior performance of traditional ML methods over GNNs is in stark contrast with results reported in recent benchmarking studies (Shchur et al., 2018; Dwivedi et al., 2020; Hu et al., 2020), highlighting the unique characteristics of network biology datasets presented and the challenges in modeling them. Below, we summarize three key challenges and the main differences from existing benchmarking graph datasets.

Challenge 1. No node features available Traditional network biology studies solely rely on biological network structures to gain insights (Barabási, Gulbahce, and Loscalzo, 2011; Barabasi and Oltvai, 2004; Köhler et al., 2008). Meanwhile, the presence of rich node features in many existing benchmarking graph datasets is crucial for the success of GNNs (Liu, Cantürk, et al., 2022), posing a challenge for GNNs in learning without meaningful node features. On the other hand, leveraging sequence and structure information of genes or gene products (e.g., proteins) can help generate useful gene features. In fact, Gu et al. (2022) (NoN) has demonstrated the power of combining structural information of proteins (as node features) with the underlying protein interaction network to enhance protein function predictions. NoN uses a graphlet-based approach to extract features from the protein graph, but more advanced GNN-based methods can be explored in future studies.

[4] https://github.com/pyg-team/pytorch_geometric/releases/tag/2.0.4

	HumanNet-DisGeNET		HumanNet-GOBP	
	APOP	AUROC	APOP	AUROC
Adj-LogReg	**1.898 ± 0.000**	0.635 ± 0.000	3.413 ± 0.000	0.756 ± 0.000
Adj-SVM	1.772 ± 0.000	0.619 ± 0.000	3.272 ± 0.000	0.721 ± 0.000
N2V-LogReg	1.651 ± 0.024	**0.652 ± 0.004**	3.383 ± 0.031	0.780 ± 0.002
N2V-SVM	1.594 ± 0.030	0.640 ± 0.002	3.230 ± 0.040	0.764 ± 0.001
LabelProp	1.874 ± 0.000	0.619 ± 0.000	3.294 ± 0.000	0.719 ± 0.000
GIN	1.621 ± 0.067	0.615 ± 0.010	3.283 ± 0.030	0.766 ± 0.004
GCN	1.756 ± 0.054	0.640 ± 0.005	3.438 ± 0.043	0.772 ± 0.004
GraphSAGE	1.718 ± 0.042	0.645 ± 0.003	**3.441 ± 0.038**	0.782 ± 0.005
GAT	1.866 ± 0.028	0.644 ± 0.003	3.387 ± 0.040	**0.787 ± 0.003**

Table 2. Model performance on two task collections using the HumanNet gene interaction network as input graph. The best-performing methods with respect to a particular metric for each group of methods (top: graph machine learning; bottom: graph neural network) are underlined; the best methods across the two groups are additionally marked as bold. The abbreviations mean the following. Adj: adjacency matrix; N2V: node2vec; SVM: support vector machines; LabelProp: label propagation.

Challenge 2. Scarce labeled data Curated biological annotations are scarce, posing the challenge of effectively training powerful and expressive models with limited labeled examples. Furthermore, a particular gene can be labeled with more than one biological annotation (multi-label setting), which is in contrast with the multi-class settings in popular benchmarking graph datasets such as Cora and CiteSeer (Shchur et al., 2018). To make a more direct comparison, we consider a specific combinations of binary labels as an effective label. Thus, despite having a decently high label rate[5] ($\sim 20\%$), the effective average number of labeled examples is only ~ 1 compared to that of existing benchmarks (20) (Shchur et al., 2018). The data scarcity issue naturally opens the door for unsupervised (Goyal and Ferrara, 2018; Cai, Zheng, and Chang, 2018) and self-supervised (Xie et al., 2022) graph learning techniques, such as *node2vec*, which partially explains the superior performance of *node2vec* in our benchmark. However, self-supervised pre-training techniques such as DGI (Velickovic et al., 2019) are still stiffled by the challenge of missing node features (see challenge 1).

Challenge 3. Dense and weighted graphs Many gene interaction networks are dense and weighted by construction, such as coexpression (Johnson and Krishnan, 2022), or by integrating multiple data sources (Szklarczyk et al., 2019). For instance, the three networks used in our benchmark are orders of magnitude denser than citation networks (Shchur et al., 2018) (Table 1a). In more extreme cases, the weighted networks can be fully connected, such as HumanBase (C. S. Greene et al., 2015). Thus, scalably and effectively learning from densely weighted gene interaction networks is an area of research to be further explored in the future.

4.2.2 Different models have their own strength

Although some methods achieve overall optimal performance on a particular dataset, no single method can achieve the best results across all individual tasks. In particular, Figure 2 shows that most methods do not correlate perfectly with each other across different tasks in the HumanNet-DisGeNET dataset, indicating that some methods may predict some particular task better than another method, and vice versa. For example, although Adj-LogReg performs the best overall on the HumanNet-DisGeNET dataset, GCN achieves significantly better predictions (t-test p-value < 0.01) for a handful of tasks (Appendix S3), such as type 2 diabetes mellitus (DOID:9352) and

[5]The ratio of the number of labeled nodes in the training set over the total number of nodes in the graph (Shchur et al., 2018)

dermatitis (DOID:2723).

In addition to overall performance, focusing on the best model for one or a few tasks of interest is more important because, in practice, experimental biologists often come with only one (or a few) gene set(s) and want to either (1) obtain new related genes or (2) reprioritize the set of genes based on their relevance to the whole gene set (Mancuso, Bills, et al., 2022). Understanding the characteristics of the task where a particular model works well is the key to make better architectural decision given a specific task of interest, and furthermore, designing new specialized GNN architectures for network biology.

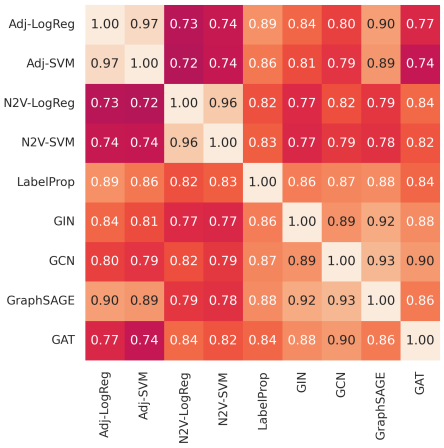


Figure 2. Pearson correlations between methods across tasks in the HumanNet-DisGeNET dataset.

5 Discussion

Apart from the challenges listed above, there are unique opportunities raised by the benchmarking datasets presented. From the biomedical perspective, predicting the role each human gene plays in orchestrating cellular processes and contributing to complex diseases is a fundamental biomedical task central to bridging the gap between genotype and phenotype. This need remains in the face of most human genes being under- or uncharacterized. Thus, most of the biomedical community around the world is conducting thousands of experimental studies to perform this task. However, experimentally characterizing function and disease associations of all 25,000 human genes is prohibitive without computational predictions narrowing down the search space. Therefore, continuous effort in developing increasingly accurate methods for predicting gene annotations is essential.

Meanwhile, from the machine learning perspective, biomedical gene annotation tasks based on gene interaction networks present an ideal benchmarking setup for graph-based ML practitioners to test new methods on real-world predictive tasks in biology and biomedicine for the following reasons:

- There are several genome-scale gene networks that vary in size, density, interaction type, and data source, along with thousands of experimentally-verified gene annotations that present a confident set of tasks for evaluating node classification and link prediction.
- These gene graphs and gene-based prediction tasks are in stark contrast to other graphs/-tasks that involve other biomedical entities, such as drug-target, drug-disease, and drug-drug interactions. There, the underlying graph edges and the prediction tasks suffer from sparse prior knowledge, making them more amenable to exploring new methods to get potentially novel findings rather than benchmarking new methods based on solid ground truth.
- Graph representation learning approaches are expected to thrive on these networks and prediction tasks because functionally related genes are likely to reside in coherent network modules, either by design (Szklarczyk et al., 2019; Hwang et al., 2019) or by the nature of gene interactions (Spirin and Mirny, 2003). This localization of functionally-related genes is a realization of the homophily network property where typical GNNs perform well (Luan et al., 2021; Zhu et al., 2020; Y. Ma et al., 2021).
- Finally, these gene-based graphs/tasks have several distinct properties from popular existing benchmarking graph datasets for node classification, such as Cora and CiteSeer. First, many gene interaction network databases combine gene interaction evidence from multi-

ple sources, resulting in much denser networks than those in other graph datasets. How GNNs scale to and perform on dense networks has yet to be systematically studied. Second, by the nature of biomedical gene annotations, the majority of tasks will have many more negative examples than positive examples, resulting in a highly imbalanced multi-label classification setting. This high-class imbalance is characteristic of most tasks in biomedicine. Although some solutions for tackling the class imbalance issue have been proposed for the multi-class setting (Buda, Maki, and Mazurowski, 2018), solutions for multi-label setting remain open-ended when, meanwhile, the effectiveness of correcting for class imbalance has been recently questioned (Goorbergh et al., 2022).

6 Future work

We plan to continually improve the `nleval` package and expand it with more data and tasks.

Heterogeneous graph objects In the current version of `nleval`, each network is represented as a homogeneous graph, where each can only have a single edge type and a single node type. Supporting multiple edge types will bring more flexibility to modeling interactions inferred by different evidence (Szklarczyk et al., 2019; Hwang et al., 2019), or modeling context-specific interactions, such as tissue (C. S. Greene et al., 2015) and cellular-compartments (Veres et al., 2015). Meanwhile, multiple node types can be helpful for modeling interactions between different types of biomolecules, such as transcription factors and non-coding RNAs (Gysi and Barabasi, 2022), or modeling cross-species networks (Law, Kale, and Murali, 2021).

Link prediction tasks Despite several large-scale and systematic efforts to map the human gene interaction network, several challenges remain to be addressed, such as the presence of false-positives (Von Mering et al., 2002) and uncharted interactions (Tabar et al., 2022). Accurately identifying high-quality interactions from high-throughput but less accurate measurements is crucial to mediate these gaps. Concretely, we will follow the link-prediction evaluation scheme proposed by a recent benchmark dataset OGBL-PPA (Hu et al., 2020) to aid the development of methods that improve the overall quality of the gene interactions.

More network data We plan to include more gene interaction networks, particularly the ones that are context-specific, such as HumanBase (C. S. Greene et al., 2015), CompPPI (Veres et al., 2015), and TissueNet (Ziv et al., 2022).

Interface with other tools

- **DGL** (M. Y. Wang, 2019) is another popular framework for graph machine learning in addition to PyG. We will provide a DGL dataset loader in the near future.
- **GLI** (J. Ma et al., 2022) is a community-driven graph dataset contribution platform with various popular benchmarking graph learning datasets. We will make several selected datasets available on GLI to further bridge the gap between the graph machine learning and the network biology communities.
- **GraPE** (Cappelletti et al., 2021) is a fast and scalable Python package written in Rust under the hood for generating node embeddings. It contains many well-known node embedding techniques such as *node2vec* (Grover and Leskovec, 2016), *LINE* (Tang et al., 2015), and *walklets* (Perozzi et al., 2017). Bringing in this toolbox will enable more comprehensive benchmarking of the baseline node embedding performance.

7 Conclusion

We presented `nleval`, an open-source Python package for rapidly setting up ML-ready benchmarking graph datasets using publicly available biological network and annotation data. `nleval` takes

care of tedious data (pre-)processing steps so that network biologists can easily set up particular datasets with their desired settings, and GRL researchers can directly use the ML-ready datasets for model development. We demonstrated and discussed the unique characteristics of network biology datasets and the challenges of modeling them using standard GNN architectures. We hope our results and package will encourage future work toward specialized GNN architectures or training strategies for network biology datasets.

Acknowledgment

This preprint was created using the LaPreprint template (https://github.com/roaldarbol/lapreprint) by Mikkel Roald-Arbøl.

References

Albert, Réka (2005). "Scale-free networks in cell biology". In: *Journal of cell science* 118.21, pp. 4947–4957.

Ali, Mehdi, Max Berrendorf, et al. (2021). "PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings." In: *J. Mach. Learn. Res.* 22.82, pp. 1–6.

Ali, Mehdi, Charles Tapley Hoyt, et al. (2019). "BioKEEN: a library for learning and evaluating biological knowledge graph embeddings". In: *Bioinformatics* 35.18, pp. 3538–3540.

Ashburner, Michael et al. (2000). "Gene ontology: tool for the unification of biology". In: *Nature genetics* 25.1, pp. 25–29.

Ata, Sezin Kircali et al. (2021). "Recent advances in network-based methods for disease gene prediction". In: *Briefings in bioinformatics* 22.4, bbaa303.

Barabasi, Albert-Laszlo and Zoltan N Oltvai (2004). "Network biology: understanding the cell's functional organization". In: *Nature reviews genetics* 5.2, pp. 101–113.

Barabási, Albert-László, Natali Gulbahce, and Joseph Loscalzo (2011). "Network medicine: a network-based approach to human disease". In: *Nature reviews genetics* 12.1, pp. 56–68.

Bard, Jonathan BL and Seung Y Rhee (2004). "Ontologies in biology: design, applications and future challenges". In: *nature reviews genetics* 5.3, pp. 213–222.

Buda, Mateusz, Atsuto Maki, and Maciej A Mazurowski (2018). "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural networks* 106, pp. 249–259.

Cai, Hongyun, Vincent W Zheng, and Kevin Chen-Chuan Chang (2018). "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 30.9, pp. 1616–1637.

Camacho, Diogo M et al. (2018). "Next-generation machine learning for biological networks". In: *Cell* 173.7, pp. 1581–1592.

Cappelletti, Luca et al. (2021). "GraPE: fast and scalable Graph Processing and Embedding". In: *arXiv preprint arXiv:2110.06196*.

Cui, Hejie et al. (2022). "On positional and structural node features for graph neural networks on non-attributed graphs". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 3898–3902.

Dessimoz, Christophe, Nives Škunca, and Paul D Thomas (2013). "CAFA and the open world of protein function predictions". In: *Trends in Genetics* 29.11, pp. 609–610.

Dixon, Scott J et al. (2009). "Systematic mapping of genetic interaction networks". In: *Annual review of genetics* 43.1, pp. 601–625.

Dwivedi, Vijay Prakash et al. (2020). "Benchmarking graph neural networks". In: *arXiv preprint arXiv:2003.0098*

Fanfani, Viola, Fabio Cassano, and Giovanni Stracquadanio (2020). "PyGNA: a unified framework for geneset network analysis". In: *BMC bioinformatics* 21.1, pp. 1–22.

Fey, Matthias and Jan Eric Lenssen (2019). "Fast graph representation learning with PyTorch Geometric". In: *arXiv preprint arXiv:1903.02428*.

Goorbergh, Ruben van den et al. (2022). "The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression". In: *Journal of the American Medical Informatics Association*.

Goyal, Palash and Emilio Ferrara (2018). "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151, pp. 78–94.

Gray, Kristian A et al. (2015). "Genenames. org: the HGNC resources in 2015". In: *Nucleic acids research* 43.D1, pp. D1079–D1085.

Greene, Casey S et al. (2015). "Understanding multicellular function and disease with human tissue-specific networks". In: *Nature genetics* 47.6, pp. 569–576.

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

Gu, Shawn et al. (2022). "Modeling multi-scale data via a network of networks". In: *Bioinformatics* 38.9, pp. 2544–2553.

Gysi, Deisy Morselli and Albert-Laszlo Barabasi (2022). "Non-Coding RNAs Improve the Predictive Power of Network Medicine". In: *arXiv preprint arXiv:2211.14800*.

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30.

Himmelstein, Daniel, Casey Greene, and Alexander Pico (2015). *Using Entrez Gene as our gene vocabulary*. DOI: 10.15363/thinklab.d34. URL: https://doi.org/10.15363/thinklab.d34.

Hu, Weihua et al. (2020). "Open graph benchmark: Datasets for machine learning on graphs". In: *Advances in neural information processing systems* 33, pp. 22118–22133.

Huang, Justin K et al. (2018). "Systematic evaluation of molecular networks for discovery of disease genes". In: *Cell systems* 6.4, pp. 484–495.

Hubbard, T et al. (2005). "Ensembl 2005". In: *Nucleic acids research* 33.suppl_1, pp. D447–D453.

Hwang, Sohyun et al. (2019). "HumanNet v2: human gene networks for disease research". In: *Nucleic acids research* 47.D1, pp. D573–D580.

Ideker, Trey and Roded Sharan (2008). "Protein networks in disease". In: *Genome research* 18.4, pp. 644–652.

Jamasb, Arian R, Pietro Lió, and Tom L Blundell (2020). "Graphein-a python library for geometric deep learning and network analysis on protein structures". In: *bioRxiv*.

Ji, Junzhong et al. (2012). "Survey: Functional module detection from protein-protein interaction networks". In: *IEEE Transactions on Knowledge and Data Engineering* 26.2, pp. 261–277.

Johnson, Kayla A and Arjun Krishnan (2022). "Robust normalization and transformation techniques for constructing gene coexpression networks from RNA-seq data". In: *Genome biology* 23.1, pp. 1–26.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.

Köhler, Sebastian et al. (2008). "Walking the interactome for prioritization of candidate disease genes". In: *The American Journal of Human Genetics* 82.4, pp. 949–958.

Law, Jeffrey N, Shiv D Kale, and TM Murali (2021). "Accurate and efficient gene function prediction using a multi-bacterial network". In: *Bioinformatics* 37.6, pp. 800–806.

Lee, Insuk et al. (2011). "Prioritizing candidate disease genes by network-based boosting of genome-wide association data". In: *Genome research* 21.7, pp. 1109–1121.

Li, Qimai, Zhichao Han, and Xiao-Ming Wu (2018). "Deeper insights into graph convolutional networks for semi-supervised learning". In: *Thirty-Second AAAI conference on artificial intelligence*.

Liu, Renming, Semih Cantürk, et al. (2022). "Taxonomy of Benchmarks in Graph Representation Learning". In: *arXiv preprint arXiv:2206.07729*.

Liu, Renming and Arjun Krishnan (2021). "PecanPy: a fast, efficient and parallelized Python implementation of node2vec". In: *Bioinformatics* 37.19, pp. 3377–3379.

Liu, Renming, Christopher A Mancuso, et al. (2020). "Supervised learning is an accurate method for network-based gene classification". In: *Bioinformatics* 36.11, pp. 3457–3465.

Luan, Sitao et al. (2021). "Is Heterophily A Real Nightmare For Graph Neural Networks To Do Node Classification?" In: *arXiv preprint arXiv:2109.05641*.

Ma, Jiaqi et al. (2022). "Graph Learning Indexer: A Contributor-Friendly Platform for Better Curation of Graph Learning Benchmarks". In: *Learning on Graphs Conference*.

Ma, Yao et al. (2021). "Is homophily a necessity for graph neural networks?" In: *arXiv preprint arXiv:2106.06134*.

Mancuso, Christopher A, Patrick S Bills, et al. (2022). "GenePlexus: a web-server for gene discovery using network-based machine learning". In: *Nucleic Acids Research*.

Mancuso, Christopher A, Renming Liu, and Arjun Krishnan (2022). "PyGenePlexus: A Python package for gene discovery using network-based machine learning". In: *bioRxiv*.

McGillivray, Patrick et al. (2018). "Network analysis as a grand unifier in biomedical data science". In: *Annual Review of Biomedical Data Science* 1, pp. 153–180.

Morris, Christopher et al. (2020). "Tudataset: A collection of benchmark datasets for learning with graphs". In: *arXiv preprint arXiv:2007.08663*.

Muzio, Giulia, Leslie O'Bray, and Karsten Borgwardt (2021). "Biological network analysis with deep learning". In: *Briefings in bioinformatics* 22.2, pp. 1515–1530.

Nelson, Walter et al. (2019). "To embed or not: network embedding as a paradigm in computational biology". In: *Frontiers in genetics* 10, p. 381.

Oliver, Stephen (2000). "Guilt-by-association goes global". In: *Nature* 403.6770, pp. 601–602.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12, pp. 2825–2830.

Perozzi, Bryan et al. (2017). "Don't walk, skip! online learning of multi-scale network embeddings". In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 258–265.

Piñero, Janet et al. (2016). "DisGeNET: a comprehensive platform integrating information on human disease-associated genes and variants". In: *Nucleic acids research*, gkw943.

Pratt, Dexter et al. (2015). "NDEx, the network data exchange". In: *Cell systems* 1.4, pp. 302–305.

Rivas-Barragan, Daniel et al. (2020). "Drug2ways: Reasoning over causal paths in biological networks for drug discovery". In: *PLoS computational biology* 16.12, e1008464.

Saito, Takaya and Marc Rehmsmeier (2015). "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one* 10.3, e0118432.

Shchur, Oleksandr et al. (2018). "Pitfalls of graph neural network evaluation". In: *Relational Representation Learning Workshop (R2L 2018), NeurIPS 2018, Montréal, Canada.*

Smith, Barry et al. (2007). "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration". In: *Nature biotechnology* 25.11, pp. 1251–1255.

Sofaer, Helen R, Jennifer A Hoeting, and Catherine S Jarnevich (2019). "The area under the precision-recall curve as a performance metric for rare binary events". In: *Methods in Ecology and Evolution* 10.4, pp. 565–577.

Spirin, Victor and Leonid A Mirny (2003). "Protein complexes and functional modules in molecular networks". In: *Proceedings of the national Academy of sciences* 100.21, pp. 12123–12128.

Stark, Chris et al. (2006). "BioGRID: a general repository for interaction datasets". In: *Nucleic acids research* 34.suppl_1, pp. D535–D539.

Szklarczyk, Damian et al. (2019). "STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets". In: *Nucleic acids research* 47.D1, pp. D607–D613.

Tabar, Mehdi Sharifi et al. (2022). "Illuminating the dark protein-protein interactome". In: *Cell Reports Methods* 2.8, p. 100275.

Tang, Jian et al. (2015). "Line: Large-scale information network embedding". In: *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077.

Velickovic, Petar et al. (2019). "Deep Graph Infomax." In: *ICLR (Poster)* 2.3, p. 4.

Veličković, Petar et al. (2017). "Graph attention networks". In: *arXiv preprint arXiv:1710.10903*.

Veres, Daniel V et al. (2015). "ComPPI: a cellular compartment-specific database for protein–protein interaction network analysis". In: *Nucleic acids research* 43.D1, pp. D485–D493.

Vidal, Marc, Michael E Cusick, and Albert-László Barabási (2011). "Interactome networks and human disease". In: *Cell* 144.6, pp. 986–998.

Von Mering, Christian et al. (2002). "Comparative assessment of large-scale data sets of protein–protein interactions". In: *Nature* 417.6887, pp. 399–403.

Wang, Minjie Yu (2019). "Deep graph library: Towards efficient and scalable deep learning on graphs". In: *ICLR workshop on representation learning on graphs and manifolds*.

Wang, Quan et al. (2017). "Knowledge graph embedding: A survey of approaches and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12, pp. 2724–2743.

Wang, Xiujuan, Natali Gulbahce, and Haiyuan Yu (2011). "Network-based methods for human disease gene prediction". In: *Briefings in functional genomics* 10.5, pp. 280–293.

Wu, Chunlei, Adam Mark, and Andrew I Su (2014). "MyGene. info: gene annotation query as a service". In: *bioRxiv*, p. 009332.

Wu, Zonghan et al. (2020). "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1, pp. 4–24.

Xie, Yaochen et al. (2022). "Self-supervised learning of graph neural networks: A unified review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu, Keyulu, Weihua Hu, et al. (2019). "How Powerful are Graph Neural Networks?" In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=ryGs6iA5Km.

Xu, Keyulu, Chengtao Li, et al. (2018). "Representation learning on graphs with jumping knowledge networks". In: *International conference on machine learning*. PMLR, pp. 5453–5462.

Yue, Xiang et al. (2020). "Graph embedding on biomedical networks: methods, applications and evaluations". In: *Bioinformatics* 36.4, pp. 1241–1251.

Zhang, Xiao-Meng et al. (2021). "Graph neural networks and their current applications in bioinformatics". In: *Frontiers in genetics* 12.

Zhou, Jie et al. (2020). "Graph neural networks: A review of methods and applications". In: *AI Open* 1, pp. 57–81.

Zhou, Kaixiong et al. (2020). "Towards deeper graph neural networks with differentiable group normalization". In: *Advances in neural information processing systems* 33, pp. 4917–4928.

Zhu, Jiong et al. (2020). "Beyond homophily in graph neural networks: Current limitations and effective designs". In: *Advances in Neural Information Processing Systems* 33, pp. 7793–7804.

Zitnik, Marinka and Jure Leskovec (2017). "Predicting multicellular function through multi-layer tissue networks". In: *Bioinformatics* 33.14, pp. i190–i198.

Zitnik, Marinka, Francis Nguyen, et al. (2019). "Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities". In: *Information Fusion* 50, pp. 71–91.

Ziv, Maya et al. (2022). "The TissueNet v. 3 Database: Protein-protein Interactions in Adult and Embryonic Human Tissue contexts". In: *Journal of Molecular Biology*, p. 167532.

A Additional information

A.1 Model parameters and settings

In this section, we briefly describe each methods and list the hyper-parameter settings we used.

A.1.1 Baseline graph machine learning methods

The first four baselines, Adj-LogReg, Adj-SVM , N2V-LogReg, and N2V-SVM, are set up by combining the node featurization schemes, Adj or N2V, and the downstream classifiers, LogReg or SVM.

- **N2V** (Grover and Leskovec, 2016) is a random-walk based node embedding method. The algorithm first generates random walk (node sequences) from the graph, and then learn to orient the nodes in the embedding space based on how often a pair of nodes co-occur within a few hops of the random walk. We used `PecanPy` (Liu and Krishnan, 2021), which is a fast and memory efficient implementation of *node2vec*, to generate the embeddings.
  - Dimension: 128
  - Number of walks per node: 10
  - Walk length: 160
  - Window size: 12
  - In-out (q) and return (p) parameters: 1
- **Adj** (Liu, Mancuso, et al., 2020) takes the rows of the adjacency matrix as the input features for each node to the downstream classifiers.
- **LogReg** and **SVM** are implemented by `scikit-learn` (Pedregosa et al., 2011)
  - Regularization: l_2
  - Regularization strength (C): 1
- **LabelProp** (Köhler et al., 2008) iteratively propagate the initial seed nodes over the graph, with probability β to transit back to the initial node. It is implemented as as part the `nleval` package.
  - Restart parameter (β): 0.9

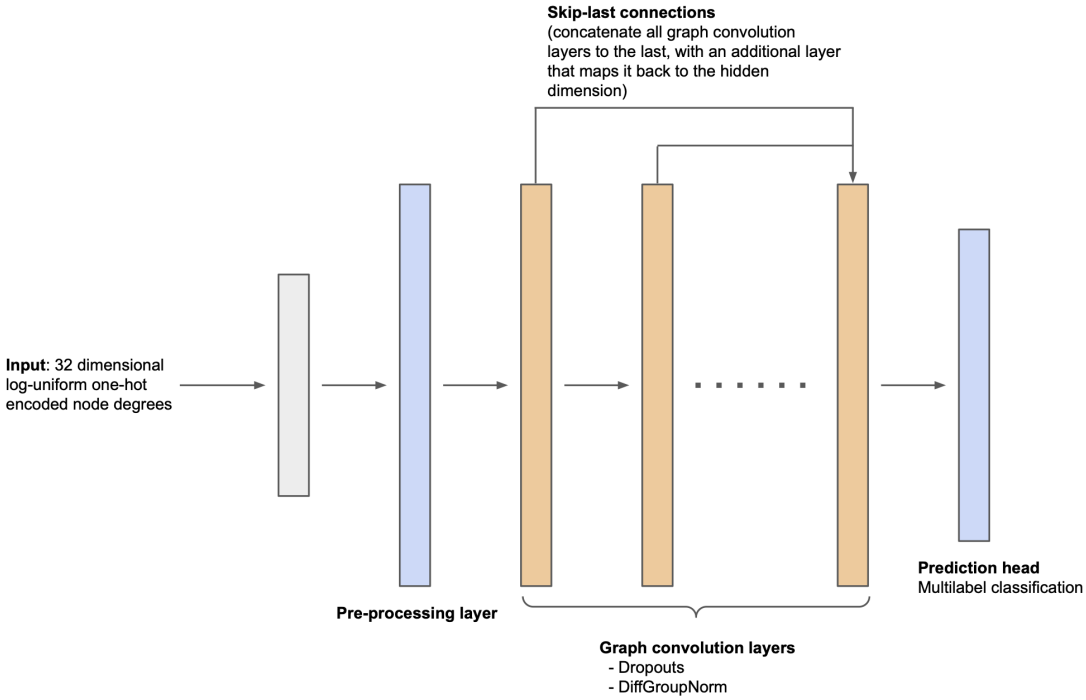
A.1.2 Graph neural networks

Node features We compute the initial features of each node by one-hot encoded node-degrees based on 32 log-uniform grids. The one-hot encoded node degrees approach has recently been shown to be a great structure encoder, whose utilization can sometimes result in performance superior to using the original node features associated with the graph (Cui et al., 2022; Liu, Cantürk, et al., 2022). The design choice of using log-uniform grids stems from the scale-free nature of biological networks (Albert, 2005). By doing so, the encodings are distributed more uniformly across nodes than using a linear grid, where most of the nodes will fall into the bin of the smallest node degree, which effectively turns the encoding into a constant.

Architecture design We set up the GNN architecture using one linear pre-processing layer with ReLU activation, followed by graph convolution layers (GCN, GIN, GAT, or GraphSAGE), and finally a prediction head layer that maps the final node representations to the predictions for each one of the multi-label classification task (Figure S1). Furthermore, for each graph convolution layer, we enable dropout to prevent the model from over-fitting early and add a differentiable group normalization layer (K. Zhou et al., 2020) to remedy oversmoothing (Q. Li, Han, and X.-M. Wu, 2018). Finally, we add skip-connections (Xu, C. Li, et al., 2018) to the graph convolution layers by concatenating the representations from each convolution layer to the end, followed by an additional linear layer to reduce the concatenated representations back to the hidden dimensions. More specific settings are listed as follow.

- Hidden dimensions: 128

- Number of convolution layers: 5
- Number of groups in group normalization: 5
- Dropout rate: 0.5
- Learning rate: 0.001
- Max epochs: 20,000
- Number of heads (GAT): 1
- Aggregation scheme (GraphSAGE): Sum



Appendix 0—figure S1. GNN architecture illustration.

A.2 The log2 fold change of average precision over the prior metric (APOP)

APOP is computed by taking log_2 of the ratio between the average precision and the prior. The prior is computed as the ratio between the number of positives and the total number of samples, which correspond to the probability that a randomly drawn sample is positive. Thus, APOP indicates how much better (> 0), or worse (< 0), a prediction is than a random guess, in two-fold change.

The average precision is a particular way to compute the **a**rea **u**nder the **p**recision **r**ecall **c**urve (AUPRC), which has been shown to be more suitable than AUROC in the case of class imbalance (Saito and Rehmsmeier, 2015; Sofaer, Hoeting, and Jarnevich, 2019). In our dataset, class imbalance is prevalent, where each class has only one or two hundred of positive examples but thousands of negative examples (Table 1b,S6).

Moreover, from a practical standpoint, making the predictions is not the full story - biologists will then try to manually analyze the top handful of predictions based on their expertise and hopefully gain biologically meaningful insights into their applications. Crucially, a human cannot analyze nearly as much data can computers can. Thus, the quality of the top few predictions is much more important than the rest, motivating the use of average precision.

B Additional tables

	BioGRID-DisGeNET		BioGRID-GOBP	
	APOP	AUROC	APOP	AUROC
Adj-LogReg	0.774 ± 0.000	0.544 ± 0.000	1.686 ± 0.000	0.621 ± 0.000
Adj-SVM	0.642 ± 0.000	0.535 ± 0.000	1.464 ± 0.000	0.595 ± 0.000
N2V-LogReg	0.743 ± 0.074	**0.558 ± 0.008**	1.876 ± 0.029	0.648 ± 0.002
N2V-SVM	0.724 ± 0.025	0.552 ± 0.005	1.770 ± 0.036	0.636 ± 0.003
LabelProp	**0.814 ± 0.000**	0.553 ± 0.000	1.248 ± 0.000	0.589 ± 0.000
GIN	0.646 ± 0.100	0.538 ± 0.011	1.637 ± 0.054	0.641 ± 0.006
GCN	0.711 ± 0.017	0.543 ± 0.004	**2.000 ± 0.028**	**0.664 ± 0.004**
GraphSAGE	0.656 ± 0.053	0.546 ± 0.006	1.928 ± 0.046	0.654 ± 0.003
GAT	0.607 ± 0.071	0.528 ± 0.014	1.780 ± 0.042	0.660 ± 0.002

Appendix 0—table S1. Model performance using BioGRID as the input graph.

	STRING-DisGeNET		STRING-GOBP	
	APOP	AUROC	APOP	AUROC
	2.604 ± 0.000	0.683 ± 0.000	3.481 ± 0.000	0.748 ± 0.000
Adj-SVM	2.596 ± 0.000	0.681 ± 0.000	3.470 ± 0.000	0.747 ± 0.000
N2V-LogReg	2.438 ± 0.037	0.733 ± 0.003	**3.728 ± 0.019**	**0.823 ± 0.002**
N2V-SVM	2.311 ± 0.067	0.720 ± 0.006	3.593 ± 0.056	0.800 ± 0.004
LabelProp	**2.616 ± 0.000**	**0.738 ± 0.000**	3.423 ± 0.000	0.787 ± 0.000
GIN	1.612 ± 0.109	0.652 ± 0.012	3.091 ± 0.090	0.770 ± 0.008
GCN	1.846 ± 0.075	0.693 ± 0.006	2.931 ± 0.135	0.769 ± 0.012
GraphSAGE	1.951 ± 0.083	0.695 ± 0.008	3.008 ± 0.207	0.769 ± 0.024
GAT	OOM	OOM	OOM	OOM

Appendix 0—table S2. Model performance using STRING as the input graph. GAT failed to execute due to running out of device memory on a 32GB V100 GPU.

Task	Adj-LogReg	GCN	Mean difference	p-value
DOID:11193	1.481 ± 0.000	3.928 ± 0.869	-2.447	4.913e-04
DOID:9352	0.534 ± 0.000	2.783 ± 0.370	-2.249	1.960e-06
DOID:2723	0.539 ± 0.000	2.692 ± 0.548	-2.153	4.973e-05
DOID:1934	0.504 ± 0.000	2.649 ± 0.367	-2.145	2.602e-06
DOID:12215	1.139 ± 0.000	3.056 ± 0.086	-1.916	7.113e-11
...				
DOID:0050735	3.913 ± 0.000	2.092 ± 0.245	1.821	4.081e-07
DOID:9500	3.706 ± 0.000	1.753 ± 0.824	1.953	1.460e-03
DOID:4079	3.230 ± 0.000	0.786 ± 0.296	2.444	1.839e-07
DOID:1247	3.334 ± 0.000	0.872 ± 0.124	2.462	1.740e-10
DOID:6419	2.798 ± 0.000	0.118 ± 0.140	2.681	2.401e-10

Appendix 0—table S3. Task specific model prediction performance on the HumanNet-DisGeNET dataset across 5 different runs. The significance of model performance difference is measured by the p-value of t-test.

Network	# nodes	# edges	Density
BioGRID	18,951	1,103,298	0.0031
BioPlex	8,044	71,084	0.0010
FunCoup	17,905	1,004,2420	0.0313
HIPPIE	17,364	768,654	0.0025
HumanNet	17,211	84,7104	0.0029
PCNet	16,968	5,047,854	0.0175
STRING	17,942	10,951,202	0.0340

Appendix 0—table S4. Statistics for all networks in `nleval` 0.1.0.

Network	# nodes	# nodes in the LCC	Discarded
BioGRID	18,958	18,951	7
BioPlex	8,252	8,044	208
FunCoup	17,905	17,905	24
HIPPIE	17,392	17,364	28
HumanNet	17,240	17,211	39
PCNet	16,968	16,968	0
STRING	17,942	16,968	0

Appendix 0—table S5. Number of genes in the original networks and number of discarded genes due to largest connected component (LCC) extraction `nleval` 0.1.0.

Label	Network	Number of tasks	Number of positives	
			avg	std
DisGeNET	BioGRID	131	210.0	130.6
	BioPlex	88	138.5	68.3
	FunCoup	132	217.6	133.9
	HIPPIE	130	210.4	128.0
	HumanNet	129	210.3	128.5
	PCNet	113	192.1	112.0
	STRING	126	213.6	130.1
GOBP	BioGRID	211	125.8	35.2
	BioPlex	73	85.6	20.7
	FunCoup	219	129.5	38.1
	HIPPIE	220	122.9	35.5
	HumanNet	204	126.2	35.5
	PCNet	247	128.8	39.7
	STRING	207	124.4	35.0
GOCC	BioGRID	69	111.2	37.8
	BioPlex	37	86.5	23.6
	FunCoup	69	114.8	40.6
	HIPPIE	69	110.6	36.9
	HumanNet	66	112.0	37.7
	PCNet	73	117.5	41.2
	STRING	68	110.2	37.5
GOMF	BioGRID	60	110.8	32.8
	BioPlex	24	73.0	16.1
	FunCoup	69	111.1	35.6
	HIPPIE	65	107.8	32.4
	HumanNet	61	111.8	31.9
	PCNet	70	115.0	35.7
	STRING	61	111.3	32.6

Appendix 0—table S6. Statistics for all datasets (network-label combinations) in `nleval` 0.1.0.