2-2022

Scalable Probabilistic Model Selection for Network Representation Learning in Biological Network Inference

K C Kishan
kk3671@rit.edu

Scalable Probabilistic Model Selection for Network Representation Learning in Biological Network Inference

by

Kishan K C

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computing and Information Sciences

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
February, 2022

Scalable Probabilistic Model Selection for Network Representation Learning in Biological Network Inference

by

Kishan K C

Committee Approval:

We, the undersigned committee members, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computing and Information Sciences.

Dr. Rui Li Dissertation Advisor	Date
Dr. Anne Haake Dissertation Co-advisor	Date
Dr. Linwei Wang Dissertation Committee Member	Date
Dr. Qi Yu Dissertation Committee Member	Date
Dr. Feng Cui Dissertation Committee Member	Date
Dr. Ricardo Figueroa Dissertation Defense Chairperson	Date

Certified by:

Dr. Pengcheng Shi Ph.D. Program Director, Computing and Information Sciences	Date

Scalable Probabilistic Model Selection for Network Representation Learning in Biological Network Inference

by

Kishan K C

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences Ph.D. Program in Computing and Information Sciences
in partial fulfillment of the requirements for the
Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

A biological system is a complex network of heterogeneous molecular entities and their interactions contributing to various biological characteristics of the system. Although the biological networks not only provide an elegant theoretical framework but also offer a mathematical foundation to analyze, understand, and learn from complex biological systems, the reconstruction of biological networks is an important and unsolved problem. Current biological networks are noisy, sparse and incomplete, limiting the ability to create a holistic view of the biological reconstructions and thus fail to provide a system-level understanding of the biological phenomena.

Experimental identification of missing interactions is both time-consuming and expensive. Recent advancements in high-throughput data generation and significant improvement in computational power have led to novel computational methods to predict missing interactions. However, these methods still suffer from several unresolved challenges. It is challenging to extract information about interactions and incorporate that information into the computational model. Furthermore, the biological data are not only heterogeneous but also high-dimensional and sparse presenting the difficulty of modeling from indirect measurements. The heterogeneous nature and sparsity of biological data pose significant challenges to the design of deep neural network structures which use essentially either empirical or heuristic model selection methods. These unscalable methods heavily rely on expertise and experimentation, which is a time-consuming and error-prone process and are prone to overfitting. Furthermore, the complex deep networks tend to be poorly calibrated with high confidence on incorrect predictions.

In this dissertation, we describe novel algorithms that address these challenges. In Part I, we design novel neural network structures to learn representation for biological entities and further expand

the model to integrate heterogeneous biological data for biological interaction prediction. In part II, we develop a novel Bayesian model selection method to infer the most plausible network structures warranted by data. We demonstrate that our methods achieve the state-of-the-art performance on the tasks across various domains including interaction prediction. Experimental studies on various interaction networks show that our method makes accurate and calibrated predictions. Our novel probabilistic model selection approach enables the network structures to dynamically evolve to accommodate incrementally available data. In conclusion, we discuss the limitations and future directions for proposed works.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisors, Dr. Rui Li and Dr. Anne R. Haake for their continuous support and dedicated involvement in every process of my Ph.D. journey. Ever since the start of my Ph.D., they allowed me to explore my research interests and supported me with utmost faith in me. Their invaluable guidance has shaped the direction of my research and taught me to be a better researcher. I would also like to thank my committee members: Dr. Feng Cui, Dr. Qi Yu, and Dr. Linwei Wang for their instructive and insightful comments. I would like to thank Dr. Pengcheng Shi for his support and guidance throughout my Ph.D. journey. I am thankful to all my collaborators and co-authors.

I would like to acknowledge my exceptional parents, Shyam KC and Ambika KC, who motivated and supported me throughout my academic journey. Their love and care have provided the backbone for my efforts. I would like to thank my younger brother, Roshan KC for his love and encouragement to continue my hard work. I would especially like to thank my wife, Sunita Bhandari for her continuous support and belief throughout my Ph.D. journey. Thank you for making countless sacrifices to help me get to this point, believing in me, and guiding me to avoid several wrong turns.

I always had the support from my close friends in Rochester, who were like a second family to me: Aayush Chaudhary, Manoj Acharya, Sushant Kafle, Sandesh Ghimire, Prashnna Gyawali, Jwala Dhamala, Kushal Kafle, Shusil Dangi, Utsav Gyawali, Krishna Neupane, Laxmi Neupane, Dilip Aryal, Bhawani Gautam, and Deep Shankar Pandey. I would like to thank them all for their invaluable impact on my progress. I would especially like to thank Sushant, Jwala, and Sandesh for their technical guidance and fruitful research discussions.

To Sunita, the love of my life

Author Publications

This dissertation is based on the following publications:

- **K C, K.**, Li R., Gilany M. (2021). Joint Inference for Neural Network Depth and Dropout Regularization. *Neural Information Processing Systems (NeurIPS).*
- **K C, K.**, Regmi, P., Li, R., Haake, A. (2021). Predicting Biomedical Interactions with Probabilistic Model Selection for Graph Neural Networks. In review.
- **K C, K.**, Li R., Cui F., Haake A. (2021). Predicting Biomedical Interactions with Higher-Order Graph Convolutional Networks. *IEEE Transactions on Computational Biology and Bioinformatics (TCBB).*
- **K C, K.**, Cui F., Li R., Haake A. (2021). Interpretable Structured Learning with Sparse Gated Sequence Encoder for Protein-Protein Interaction Prediction. *International Conference on Pattern Recognition (ICPR).*
- **K C, K.**, Li R., Cui F., Yu, Q., Haake A. (2019). GNE: A deep learning framework for gene network inference by aggregating biological information. *BMC Systems Biology.*
- **K C, K.**, Li R., Cui F., Haake A. (2018). Learning topology-preserving embedding for gene interaction networks. *European Conference on Computational Biology (ECCB).*

The author has further contributed to the following publications:

- **K C, K.**, Subramanya, S. K., Li R., Cui, F. (2021). Machine learning predicts nucleosome binding modes of transcription factors. *BMC Bioinformatics.*
- Yin P., **K C, K.**, Shi W., Yu, Q., Li, R., Haake, A., Miyamoto, H., Cui, F. (2020). Histopathological distinction of non-invasive and invasive bladder cancers using machine learning approaches. *BMC Medical Informatics and Decision Making (MIDM).*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A biological system is a complex network of heterogeneous molecular entities such as genes, proteins, and other biological molecules linked together by their interactions. These biological entities interact with each other via direct or indirect associations and contribute to various biological characteristics of the biological system [8, 9, 10]. The study of these molecular entities and their interactions not only plays a crucial role in understanding biological phenomena but also provides insights into the molecular etiology of diseases as well as the discovery of putative drug targets [11].

The biological systems represent complex dynamical systems, where the interactions at a local level give rise to the emergent dynamics on the system's level [12, 13]. Specifically, the combination of positive and negative feedback between heterogeneous molecular entities within a biological system results in the non-linear dynamics of the system. The information about interactions between these heterogeneous entities plays a crucial role to study the non-linear dynamics of the biological system. However, the known interactions between the heterogeneous molecular entities are far from complete due to experimental limitations and unsupervised construction from the omics dataset, *i.e.*, the interactions between entities are noisy, sparse, and incomplete. Such missing interactions limit the ability to create a holistic view of the biological system and have a system-level understanding of biological phenomena.

Over the past decade, there have been huge technological advancements in high-throughput technologies that have greatly increased the ease and significantly reduced the cost of generating bio and medical data, resulting in ever-increasing amounts of omics data. During the same time, the significant improvement in computational power and the development of novel methodologies capable of handling large datasets provide us with the platform for analyzing these large heterogeneous

datasets to understand the biological system [14]. Therefore, there is a need for the development of effective computational methods that exploit available computational power to analyze heterogeneous datasets and infer novel interactions between heterogeneous molecular entities in the system.

1.1 Motivating challenges

For the system-level understanding of the biological phenomena, it is crucial to know if interactions are missing or novel interactions are likely to be identified in the future. Since the available information about interactions within the biological system is noisy and incomplete, predicting interactions using heterogeneous datasets is a challenging yet important problem in biology. In this dissertation, we term the problem as "**Biological Network Inference**".

Biological network inference involves the design of appropriate computational methods for extracting features from heterogeneous data to represent biological entities and modeling their interactions using extracted features. Various computational methods have been proposed over the past decades to predict interactions between biological entities [15, 16, 17, 18, 19]. However, the interaction inference still suffers from several unresolved challenges. We summarize the major challenges associated with the biological network inference as follows:

- **Modeling known interactions**: A network constructed from known interactions between biological entities informs about the explicit structure of the biological system and computational models should make use of and learn to extract information represented in the form of a network. A large number of publicly available databases such as BioGRID, DisGeNET provides information for known interactions between heterogeneous entities. The main challenge in machine learning is to extract information about interactions between heterogeneous entities from the constructed network and to incorporate that information into the machine learning model. Since classic machine learning methods rely on summary statistics or hand-engineered features such as centrality measures, degree distribution, average path length, diameter, clustering coefficients, subgraphs, network motifs. These extracted patterns only provide limited coverage about interactions.

- **Integrating heterogeneous datasets**: Since the factors necessary to understand biological phenomena cannot be captured by a single data type, several machine learning approaches for biological network inference that are only capable of examining specific data types fail to explain such complex phenomena. This calls for methods capable of integrating heterogeneous

datasets to gain a comprehensive understanding of a biological phenomena [20]. However, there are several challenges associated with the integration of heterogeneous data. First, biological data are high-dimensional but sparse. Second, these data are often incomplete and biased to certain aspects because of the nature of measurement technology [21], and investigative biases [22].

- **Locality problem:** Most biological characteristics of the system arise from the complex interactions between heterogeneous molecular entities [23]. Also, the effect on one molecular entity is propagated via pathways *i.e.* series of interactions among other molecules in the system to form a certain product or a change in a cell. It is thus crucial to consider higher-order network information along with local network properties to understand the structure and behavior of the complex networks. However, various methods only consider the local neighborhood information such as direct neighbors and fail to capture the global structure of the network.

- **Heuristically designed neural architectures**: Recently, neural network approaches have shown great success in predicting the likelihood of interaction between two molecular entities using omics dataset [19, 24]. However, such methods rely on the heuristically designed neural network structures with manually tuned parameters: number of layers (*i.e.* depth) and number of neurons (*i.e.* width). Although pre-determined network structures provide good performances, deep networks are prone to overfitting and also tend to be poorly calibrated with high confidence on incorrect predictions [25, 26].

1.2 Scope and research questions

In this dissertation, we are primarily focused on developing general computational approaches for link prediction with an application to biological network inference. To this aim, we evaluate our novel methods across diverse datasets to demonstrate their effectiveness and also investigate their efficiency. Starting with small and well-studied datasets such as E. coli and yeast, we explore the applicability of our method on various datasets such as human protein-protein interactions, human gene-disease associations, with an increased number of nodes and edges. If the number of nodes in the network is large, it has sparser connectivity which leads to computational challenges such as scalability, and the ability to handle sparse data. The focus of this research is structured in two parts as summarized in Figure 1.1.

In Part I, we construct a biological network using known interactions to represent the structure of

the biological system. We then introduce network representation learning approaches that make use of explicit network structure and integrate other "omics" data for biological interaction prediction. We focus on designing neural network structures for network representation learning to predict the probability of missing interactions.

In Part II, we present a unified Bayesian model selection method to jointly infer the most plausible neural network depth warranted by data and modulate neuron activations via dropout regularization simultaneously. In particular, to infer network depth we define a beta process over the number of hidden layers which allows it to go to infinity. Layer-wise activation probabilities induced by the beta process modulate neuron activations via binary vectors of a conjugate Bernoulli process. We extend the joint inference framework to infer neural network depth for biological network inference.

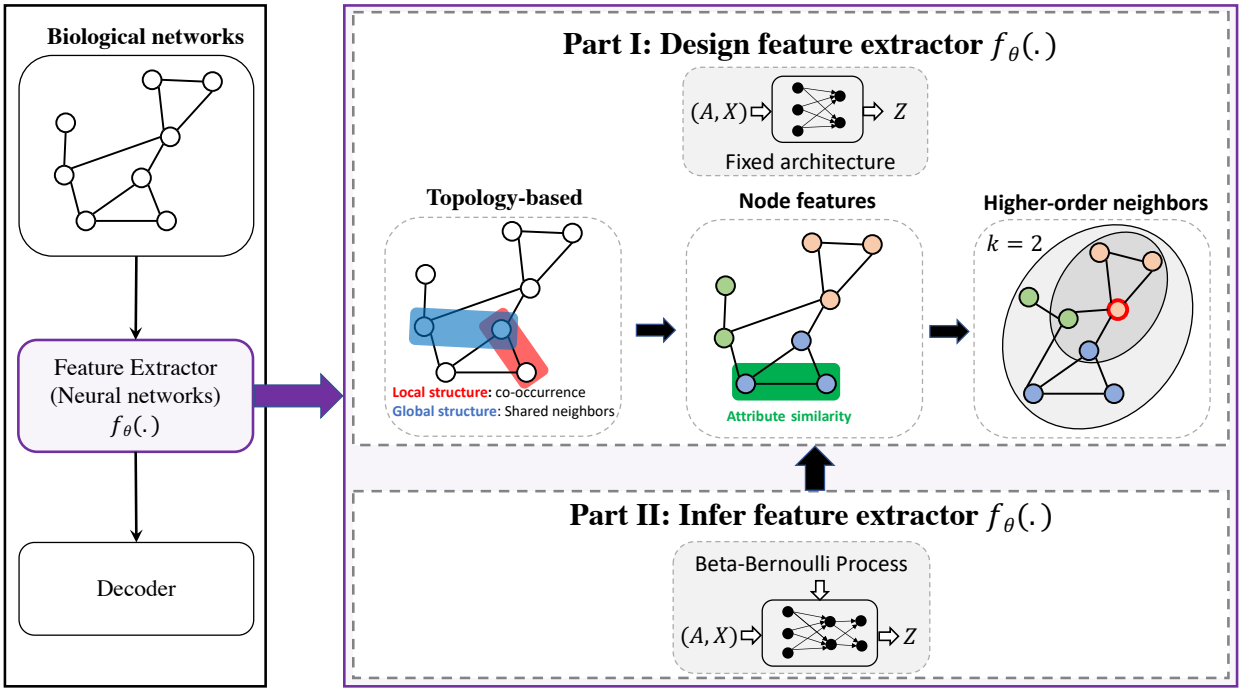


Figure 1.1: Research focus of this dissertation.

In this dissertation, we aim to answer the following research questions:

Q1: ***Are topological properties of the biological network constructed from known interactions sufficient to predict missing interactions?***

We introduce BioNetEmbedding, a novel topology-preserving embedding model that captures the topological properties of the biological network in Chapter 3. By preserving network properties,

the model learns continuous representation for entities in the network such that the entities with similar topological patterns are placed closer to each other in the latent representation space. The continuous representation of biological entities significantly simplifies downstream tasks such as interaction prediction. We demonstrate an application of the proposed model on biological interaction networks with thousands of nodes and hundreds of thousands of edges. It improves upon earlier work on network embeddings such as Isomap, node2vec, and LINE. Furthermore, we apply our proposed model to biomedical networks such as drug-drug interactions, gene-disease associations, drug-target interactions and show that our model achieves significant improvements over 11 network representation learning approaches.

Q2: ***How can we incorporate additional information with network information to improve biological interaction inference?***

To answer this question, we review various approaches to learn representation by integrating a network structure with additional information and realize that these models are unable to incorporate additional information for network representation learning and fail to generalize well to nodes with few interactions.

In Chapter 4, we extend BioNetEmbedding to address this question: Gene Network Embedding (GNE). GNE learns the representations to unify known genetic interactions and gene expression data for genetic interaction prediction. We demonstrate the application of this model to the task of temporal link prediction. Furthermore, a set of novel gene interaction predictions are validated by up-to-date literature-based database entries.

Moreover, we propose a novel interpretable deep learning framework to incorporate protein sequences with network information in Chapter 5. Specifically, we integrate a structured and sparse regularization with a sequence encoder to learn representations of the sequences and demonstrate its application on protein-protein interaction prediction. We also show that our model generates sparse masks that align with biologically interpretable motifs from Pfam database [27].

Q3: ***Are higher-order neighborhood properties informative for interaction prediction?***

To aggregate neighborhood features at various distances, we introduce a deep method with a novel higher-order graph convolution (HOGC) encoder [28] and a novel bilinear decoder for link prediction in Chapter 6. Our proposed method aggregates the features of neighbors at various distances and learns the linear mixing of these features to obtain the informative representations of

biological entities. We evaluate our method on various interaction networks such as protein-protein, drug-drug, drug-target, and gene-disease interactions and show that it achieves more accurate and calibrated predictions.

Q5: ***Does the neural network structure with inferred depth and neuron activations provide benefit over heuristically designed network structure for biological interaction prediction tasks?***

The design and development of neural network structures heavily rely on expertise and experimentation. Although these models have achieved promising performance, the heuristically designed structure might be a sub-optimal one and are also prone to overfitting. Furthermore, these predetermined network structures cannot grow accordingly as more data are observed and requires retraining neural network structures from scratch. Thus, in Chapter 7, we propose a unified Bayesian model selection to infer the most plausible neural network depth warranted by data and perform dropout regularization simultaneously. Furthermore, we extend the framework to infer the depth of graph neural networks for biological network inference in Chapter 8.

1.3 Outline

To provide the readers with essential background knowledge, Chapter 2 provides a quick introduction to biological networks and discusses related works on deep neural networks to learn the representation of these networks. It further discusses the problem of overfitting in deep learning, different approaches to avoid the overfitting problem, and tuning of neural network hyperparameters.

In Part I, we propose novel deep neural network structures to model biological interactions from heterogeneous biological data. In particular, we present a framework to learn representation for biological entities by capturing local neighborhood information in Chapter 3. In the subsequent chapters, we will extend this framework to incorporate additional information with network structure. Specifically, in Chapter 4, we discuss our method for integrating continuous features such as gene expression data with gene network structure. Next, Chapter 5 integrates sequential attributes such as protein sequences with network structure for prediction of protein-protein interaction. Chapter 6 introduces a novel approach to consider higher-order network information along with local neighborhood and additional features for biological link prediction.

In Part II, we propose a Bayesian model selection to jointly infer the most plausible neural network depth and dropout regularization simultaneously in Chapter 7. We further extend the framework to infer the depth for graph neural networks for biological network inference in Chapter 8.

Chapter 2

Background

This chapter aims to provide a quick overview of biological networks, the sources of these networks, and several related works that are extensively used throughout this dissertation. Furthermore, we also discuss the design of deep neural network structures, the problem of overfitting and overconfident predictions for deep networks, and various approaches to address these issues.

2.1 Biological networks

A natural way to represent the complex interactions between heterogeneous molecular entities in the biological system is in form of a network, where the molecular entities are represented as nodes and their pairwise relationships as edges [29]. The network representation of a biological system provides a conceptual and intuitive framework to investigate and understand the interactions between different molecular entities in a biological system. The ultimate aim of network representation is to provide a hypothesis to understand the cellular organization and also inform how the perturbations on one entity propagate in the system. Furthermore, analyzing the topological properties of the biological networks can help to uncover previously unknown relationships, identify the pathways, and the important modulators in the networks.

Specifically, network representation of the biological system involves two important steps: (1) identify the critical entities of the biological system (nodes) and (2) the nature of the interactions between these entities (edges) [30]. The information about entities and their interactions comes from various sources of data, describing various facets of the biological system:

- **High-throughput datasets**: Various high-throughput technologies have generated large amounts of data that provide information about the interaction between various entities in the biological system. For example, the networks of proteins, and their physical interactions [31] also known as protein-protein interactions can be obtained via yeast two-hybrid, affinity purification, co-immunoprecipitation.
- **Literature text-mining**: The biomedical literature can serve as a resource to extract information about interconnected proteins through their coexistence in the literature [32]. Specifically, text-mining-based approaches search for statistically significant co-occurring biological entities that can greatly increase the coverage of the biological networks.
- **Manual curation of scientific literature**: Scientific curators or domain experts extract the information from the published scientific papers and store them in the database. However, the curation process is expensive and time-consuming.
- **Computational predictions**: Various computational methods have been proposed to make use of existing experimental pieces of evidence to predict novel interactions/relationships between biological entities. Such methods provide lab-testable hypotheses by identifying novel interactions between biological entities and also refine the space of experimentally derived interactions.

Some of the most common types of biological networks are gene regulatory networks (GRNs), protein-protein interaction (PPIs) networks, drug-target interaction networks (DTIs), and gene-disease interaction networks (GDIs).

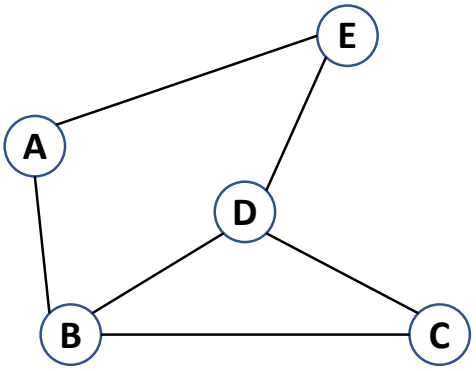
We can formally define a biological network as:

Definition 2.1. ***(Biological network)****: A biological network is defined as* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ *where* $\mathcal{V}$ *denotes the set of nodes representing biological entities such as proteins, genes, drugs, diseases and* $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$ *denotes the set of interactions between these entities.* $\mathbf{X}$ *represents additional information such as high-throughput functional genomics data. The interaction between biological entities corresponds to physical binding for protein-protein interaction, regulatory relationships for gene-gene interactions, the association between genes and diseases for gene-disease interactions, binding of a drug to a target location such as proteins, genes for drug-target interactions.*

In a biological network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, the edge $e_{ij} \in \mathcal{E}$ is associated with an interaction score $s_{ij} \geq 0$ indicating the strength of the connection between entities v_i and v_j. If entities v_i and v_j is not linked by an edge, $s_{ij} = 0$. We name interactions with $s_{ij} > 0$ as positive interactions and $s_{ij} = 0$

as negative interactions. In this dissertation, we consider score s_{ij} to be binary, indicating whether genes v_i and v_j interact or not. We later define s_{ij} as first-order proximity in Definition 3.1.

We express the biological network mathematically in the form of an adjacency matrix $\mathbf{A}$. The elements of $\mathbf{A}$ are binary values of $\{0, 1\}$, indicating the presence or absence of the edges in the network. Figure 2.1 shows the adjacency matrix for undirected network.



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	0
C	0	1	0	1	0
D	0	1	1	0	1
E	1	0	0	1	0

Figure 2.1: Biological network and its adjacency matrix representation.

2.2 Network representation learning (NRL)

Networks are becoming ubiquitous in real-world applications and have been attracting increasing attention in recent years. The analysis of these networks plays a crucial role in a variety of applications across multiple domains. For example, social networks are analyzed to classify users into meaningful social groups, enabling different downstream tasks such as user search, friendship, and content recommendation, and targeted advertising; in biological networks, inferring interactions between proteins can facilitate the drug discovery process to treat diseases.

Network representation learning (NRL) aims to learn latent lower-dimensional representations of entities in the network while preserving network structure and additional information about nodes. We can formally define biological network representation learning as:

Definition 2.2. ***(Biological network representation learning)****: Given a biological interaction network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with a set of known interactions in $\mathcal{E}$, additional information about biological entities in $\mathbf{X}$, the task of network representation learning aims to learn a mapping function f that*

maps the network structure and the additional information to a latent d-dimensional space $\mathbf{Z}$*;* $v \rightarrow \mathbf{z}_v \in \mathbb{R}^d$*. The function* f *preserves the original network information such that two nodes with similar topological structures and attributes should be represented similarly in the learned representation space.*

The resulting latent low-dimensional representation of biological entities should satisfy the following criteria:

- **Low dimensionality**: The dimension d of the representation space should be much smaller $d \ll |\mathcal{V}|$, providing memory efficiency and scalability of downstream network analysis tasks.
- **Informative**: The learned latent representation preserves the proximity between the entities in the original network, reflected by the network structure and additional information.
- **Continuous**: The continuous representation of the entities supports the use of off-the-shelf machine learning algorithms for various downstream tasks such as entity classification, clustering, and link prediction.

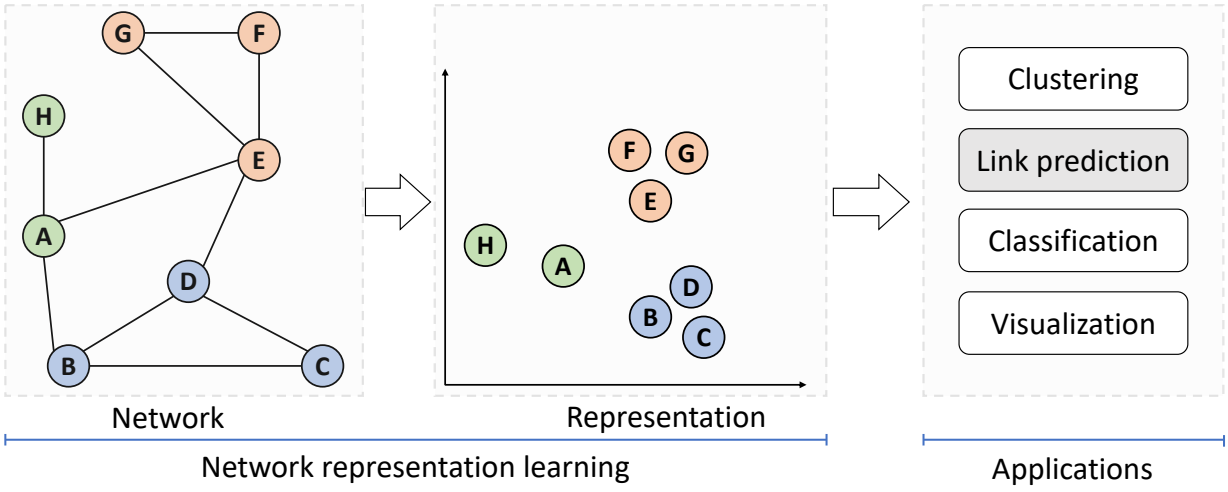


Figure 2.2: An overview of network representation learning.

Figure 2.2 shows the overview of network representation learning and a variety of downstream tasks. In this dissertation, we focus on link prediction in a biological network. However, our approach can be easily extended to other tasks. Next, we formally define the task of interaction prediction as:

Definition 2.3. ***(Interaction prediction)****: Given a biological interaction network* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ *and the set of potential biological interactions* $\mathcal{E}'$*, we aim to learn an interaction prediction model* $\mathbf{f}$ *to predict the interaction probabilities of* $\mathcal{E}'$*,* $\mathbf{f} : \mathcal{E}' \rightarrow [0, 1]$*.*

Additional definitions for network representation learning will be discussed in respective chapters.

2.3 Encoder-decoder perspective for NRL

We follow the encoder-decoder framework of NRL [33] to discuss various methods. The encoder-decoder framework relies on the idea that if we can decode the graph information in the original network from the encoded low-dimensional representations, then in principle, the representations contain all information necessary for the downstream tasks. Encoder-decoder view of NRL includes four methodological components:

- **Proximity score** $s_{ij} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$ to measure how closely connected two nodes are in the network. In this dissertation, we define $s_{ij} = A_{ij}$ such that adjacent nodes have proximity of 1, otherwise 0.
- **Encoder** $f_{ENC}(v_i)$ to learn the representation $\mathbf{z}_i$ of the entities of the network using the network structure and additional information. The encoder contains trainable parameters that are optimized via training.
- **Decoder** $f_{DEC}(\mathbf{z}_i, \mathbf{z}_j)$ to reconstruct the pairwise proximity $\hat{\mathbf{A}}_{ij}$ between entities from the learned representations $\mathbf{Z}$. The decoder function may or may not have trainable parameters. For example, matrix factorization approaches use inner-product decoder (i.e. $\mathbf{z}_i^T \mathbf{z}_j$), and end-to-end deep learning models use parameterized pairwise decoder to decode the probability of link existence [19].
- **A loss function** which determines how the reconstructed proximity scores $f_{DEC}(\mathbf{z}_i, \mathbf{z}_j)$ matches with true proximity values s_{ij}.

The definition of these four components differs between various NRL methods. Encoder-decoder framework can be summarized in Figure 2.3.

2.4 Common methods for NRL

Learning to encode the original network structure to latent representation space is a long standing problem [15, 18, 33, 34]. Various methods have been proposed to address different objectives by incorporating different types of information (Figure 2.4).

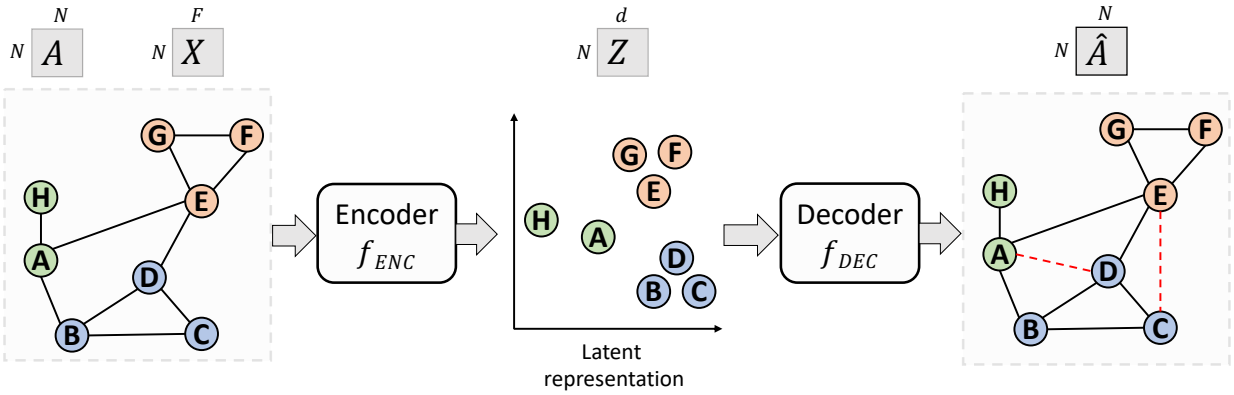


Figure 2.3: Encoder-decoder NRL framework for link prediction. NRL methods encode network structure $\mathbf{A}$ and features $\mathbf{X}$ to obtain latent representation $\mathbf{Z}$ and pairwise proximity $\hat{\mathbf{A}}$ is reconstructed from $\mathbf{Z}$. Red dashed lines represent potential interactions.

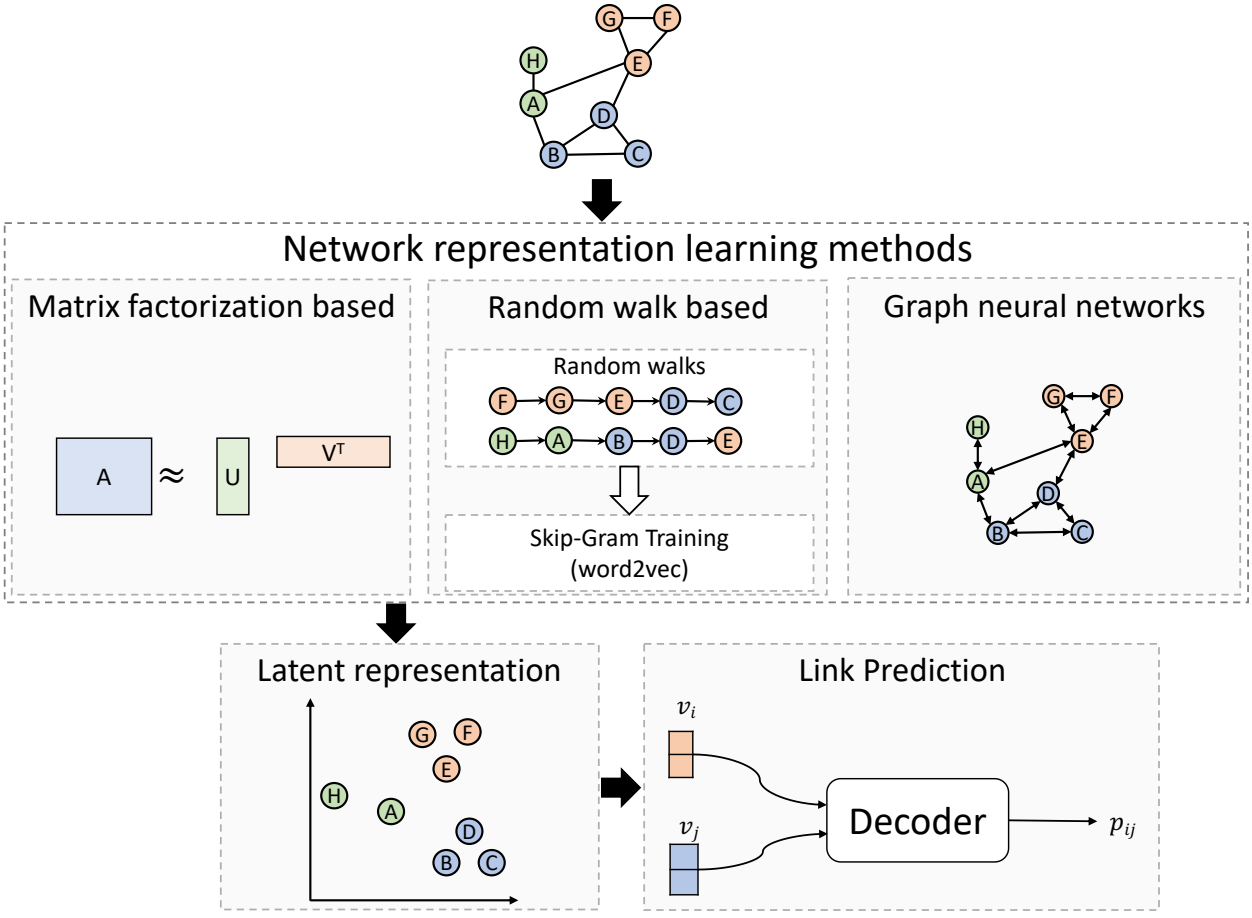


Figure 2.4: Pipelines for NRL methods for link prediction

2.4.1 Matrix factorization

An adjacency matrix is a common representation of the topology of the network as shown in Figure 2.1, where rows and columns correspond to the entities in the network and the entries in the matrix indicate their interactions. A N-dimensional row vector or a column vector from the adjacency matrix can be used to represent the entity, where N is the number of entities in the network. Matrix factorization (MF) methods are studied as dimensionality reduction techniques and aim to factorize the adjacency matrix into lower-dimensional matrices to represent the network in N-dimensional space. The manifold structure and topological properties hidden in the original network structure are preserved in the low-dimensional space. Various MF methods that have been applied to solve NRL tasks are Isomap [16], Locally Linear Embedding (LLE) [35], Laplacian Eigenmaps (LEs) [36], Singular value decomposition (SVD), graph factorization (GF) [37]. Recent methods for matrix factorization such as GraRep [34] and HOPE [38] focus on high-order proximity between network entities to preserve the global graph structure.

2.4.2 Random walk

Building upon the promising work on learning word representations from sentences [39], random walk-based methods generate the sequences of nodes through random walks on graphs and the word2vec model is used to learn the representation for nodes in the network. Such representation can preserve the structural and topological properties of the network.

DeepWalk [40] is the pioneering work in this direction that generates the truncated random walks on a network. Improving on deepwalk, node2vec uses a biased random walk by combining breadth-first sampling with depth-first sampling to generate node sequences. Furthermore, struc2vec [41] models the structural identity of biological entities in the network with an assumption that nodes with similar network structures may perform similar functions.

2.4.3 Graph neural networks

NRL, by definition, aims to learn a mapping function to encode the entities from the original network space to a latent low-dimensional vector space. NRL was originally studied with matrix factorization approaches as dimensionality reduction of the original network represented by adjacency matrix A. Since the formation process of the network is highly non-linear, the linear mapping

assumption of matrix factorization approaches may not be adequate to learn the appropriate mapping function.

With the recent success of neural network models in various fields such as computer vision, natural language processing, neural networks for NRL has also gained significant interest. Variants of neural network models such as multilayer perceptron (MLP) [18], autoencoder [42], graph convolutional network (GCN) [43, 44] have been explored for these tasks. The use of network structure information depends on the choice of neural network architecture. In particular, LINE [18] applies a single-layer MLP model to learn vector representation by approximating first-order and second-order proximity between entities in the biological network. DNGR [42] applies stacked denoising autoencoders on positive pointwise mutual information (PPMI) matrix to learn low dimensional representation. Similarly, SDNE [45] adopts Laplacian eigenmaps (LEs) to preserve first-order proximity and deep autoencoder to reconstruct neighborhood structure for each node to preserve second-order proximity.

Recently, graph neural networks (GNNs) have shown great success in modeling network-structured data [43, 44, 46]. GNNs have demonstrated their ability to recursively incorporate information from neighboring nodes in the network, naturally capturing both network structure and node attributes. Graph autoencoder (GAE) [43] applies graph convolutional encoder to capture network structure and node attributes by aggregating information from direct neighbors and employs inner product decoder to reconstruct adjacency matrix representing the network structure. Unlike GAE, SkipGNN [19] proposes to incorporate features from direct and indirect neighbors for biological link prediction. Although there has been promising development of GNN models for node classification across various domains, GNN methods for biological link prediction have been largely unexplored. Furthermore, various GNN models lack interpretability and thus fail to generate explanations for predictions, limiting their applicability in biological problems.

2.5 Design and inference of neural network structures

One crucial aspect of the deep learning progress that we have witnessed over the past decade is novel neural network structures. The design of the complex neural network structure plays an important role in learning the feature representation of data and the performance on the final task. However, overly complex neural structures are prone to overfitting and may capture spurious correlations from the data. The design of such complex structures relies heavily on prior knowledge and experience, which is a time-consuming and error-prone process. Moreover, deep networks tend

to be poorly calibrated with high confidence on incorrect predictions.

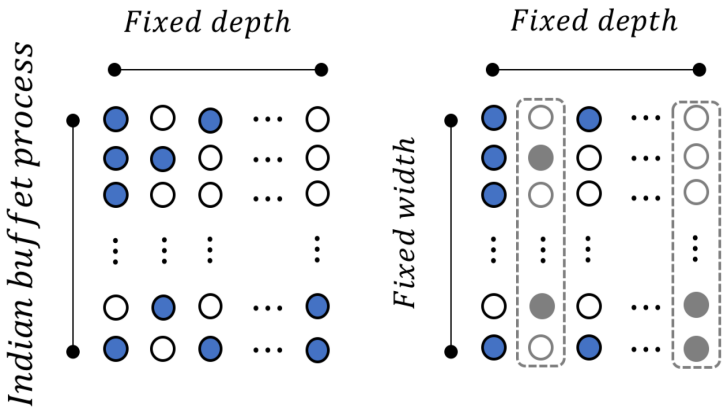


Figure 2.5: Demonstrations of a dropout variant (left), and structure selection methods (right). The dropout variant defines an Indian buffet process per layer to infer width, but depth is fixed [1, 2]. Structure selection methods can only reduce depth of a pre-determined network structure [3,4,5,6].

Neural network structures are pre-determined based on prior knowledge and experience with an expectation that the pre-determined structure results in a good performance. It requires the depth (the number of hidden layers) and width (number of neurons in each layer) to be carefully set. With the choice of these parameters comes the challenge of the network being prone to overfitting. Various regularization approaches have been proposed to prevent overfitting caused by the great flexibility of deep neural networks. These methods can be broadly classified into two groups:

2.5.1 Dropout and its variants

Dropout is an effective neural network regularization technique [47, 48]. By randomly pruning neurons and their connections from a network structure with a dropout rate during training, it can prevent DNNs from overfitting [49,50,51]. [52] interprets dropout from a Bayesian perspective to quantify uncertainty. Variational dropout proposes a stochastic gradient variational inference approach to learn the dropout rate from data with a constraint on large dropout rate values [53]. [54] extends the variational dropout to set unbounded individual dropout rate per weight. Concrete dropout proposes a continuous relaxation of the dropout's discrete masks to automatically tune the dropout probability, and obtain its uncertainty estimates [55]. Bayesian nonparametrics are also applied to extend the variational dropout approaches by defining an Indian buffet process (IBP) over neurons per hidden layer [1] or over channels in CNNs [2] to infer network widths, as in Figure 2.5 (left). Another body of work leverage the non-parametric Bayesian framework to infer the

dimensionality of latent representation encoded by DNNs [56, 57].

2.5.2 Structure selection methods

Structure selection methods improve training efficiency by reducing the depth of a pre-arranged network structure, as in Figure 2.5 (right). Some lead to well-performed smaller networks, and others achieve better uncertainty calibration. To build smaller models that perform just as well, [3] proposes to reduce a large neural network by merging adjacent hidden layers with only linear relationships being activated in between. A stochastic depth method randomly drops a subset of hidden layers from a large network by bypassing them with an identity function in each mini-batch to speed up the training session, but it still deploys the large network structure at test time [4]. [6] shows that the multiplicative noise from dropout induces structured shrinkage priors over a neural network's weights, and they further extend the shrinkage framework based on ResNet structures [58] to model the probabilities of hidden layers being used. In particular, the work also indicates that the framework is equivalent to the stochastic depth regularization [4]. [5] proposes to tune a hidden layer's influence on a prediction by learning a bypass variable between adjacent-layer connections and skip connections with a variational Bayes method. The method also prunes neurons separately. [7] proposes a Bayesian model averaging method to down-weight deeper layers by directly connecting every hidden layer to the output layer, and achieve competitive performances with uncertainty calibration. [59] achieves model effectiveness and computational efficiency by inferring a distribution of connections and units in the context of local winner-takes-all DNNs with IBP.

Part I

Network Representation Learning for Biological Interaction Prediction

Chapter 3

Representation learning for biological networks using local network properties

Understanding the functional aspects of genes or proteins is crucial to providing insights into underlying mechanisms for different health and disease conditions. As discussed in Section 2.1, the known genetic interactions between genes or proteins in the biological system can be represented as a gene interaction network. Gene interaction networks model the complex biological phenomena in the biological system and provide rich information to infer functional relationships among genes. However, a comprehensive representation of topological properties of gene interaction networks, to achieve a more accurate inference of novel genetic interactions, remains a challenge.

In this chapter, we describe a deep neural network architecture to learn the lower-dimensional representation for each gene, by preserving direct and indirect topological proximity between genes, that characterizes the topological context of each gene (Figure 3.1). These representations can be plugged into off-the-shelf machine learning methods to derive deeper insights into the structure of gene interaction networks and also functional insights about genes. In this chapter, we discuss the model that trains on the biological network only. We evaluate the proposed model on diverse datasets including small and well-studied E. coli and yeast datasets with large number of known interactions readily available from public databases as well as larger datasets to demonstrate its effectiveness and scalability.

3.1 Preliminaries

We formally define the problem of gene network inference as a network embedding problem using the concepts of network proximity as demonstrated in Figure 3.1.

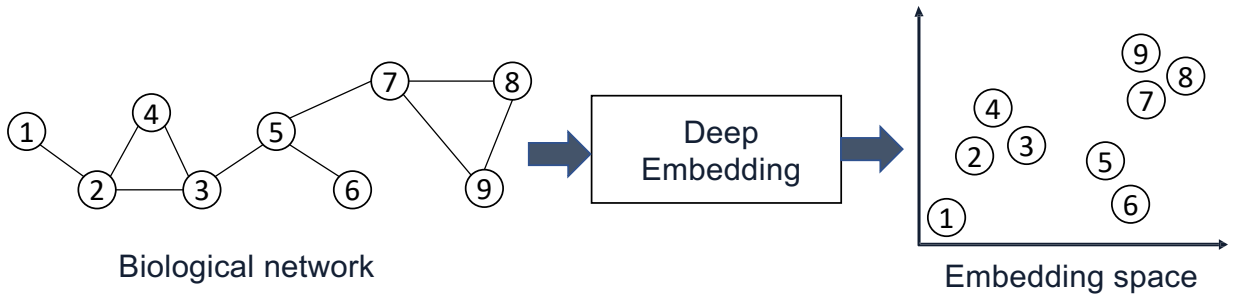


Figure 3.1: A overview of topology-preserving embedding.

According to Definition 2.1, gene network is a biological network with genes or proteins as nodes and their interactions as edges. The interaction between genes corresponds to either a physical interaction through their gene products, e.g., proteins, or one of the genes alters or affects the activity of other genes. Genes directly connected with a gene v_i in the gene network denote the local network structure of gene v_i. We define local network structures as the first-order proximity of a gene.

Definition 3.1. ***(First-order proximity)****: The first-order proximity in a gene network is the pairwise interactions between genes. Score s_{ij} indicates the first-order proximity between gene v_i and v_j. If there is no interaction between gene v_i and v_j, their first-order proximity s_{ij} is 0.*

Genes are likely to be involved in the same cellular functions if they are connected in the gene network. On the other hand, even if two genes are not connected, they may be still related to some cellular functions. This indicates the need for an additional notion of proximity to preserve the network structure. Studies suggest that genes that share a similar neighborhood are also likely to be related [60]. Thus, we introduce second-order proximity that characterizes the network neighborhood of the genes.

Definition 3.2. ***(Second-order proximity)****: Second-order proximity denotes the similarity between the neighborhood of genes. Let $\mathbb{N}_i = \{s_{i,1}, \ldots, s_{i,i-1}, s_{i,i+1}, \ldots, s_{i,M-1}\}$ denotes the first-order proximity of gene v_i, where $s_{i,j} > 0$ if there is direct connection between gene v_i and gene v_j, oth-*

erwise 0. Then, the second order proximity is the similarity between $\mathbb{N}_i$ *and* $\mathbb{N}_j$*. If there is no path to reach gene* v_i *from gene* v_j*, the second proximity between these genes is 0.*

As defined in Definition 2.2, we propose to encode genes to low-dimensional vectors $\mathbf{z} \in \mathbb{R}^d$ that preserves their network proximities in the original network.

3.2 Biological network embedding

The proposed deep learning framework as shown in Figure 3.2 utilizes gene network structure to learn a latent representation for the genes. Embedding of a gene network projects genes into a lower dimensional space, known as the embedding space, in which each gene is represented by a vector. We list the variables to specify our framework in Table 3.1.

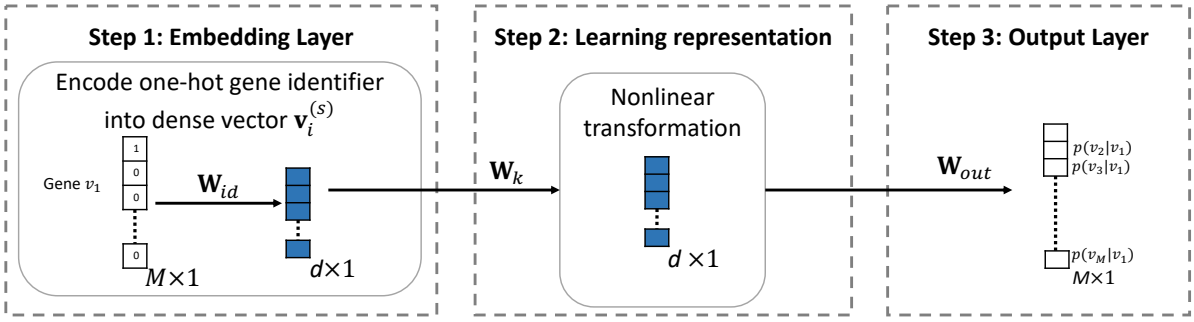


Figure 3.2: Overview of topology-preserving embedding framework for genetic interaction prediction

Gene network modeling

The proposed framework preserves the first-order and second-order proximity of genes in the gene interaction network. The key idea of network structure modeling is to estimate the pairwise proximity of genes in terms of the network structure. If two genes are connected or share similar neighborhood genes, they tend to be related and should be placed closer to each other in the embedding space. Inspired by the Skip-gram model [39], we use one-hot encoding to represent a gene. Each gene v_i in the network is represented as an M-dimensional vector where only the i^{th} component of the vector is 1.

Symbol	Definitions
M	Total number of genes in gene network
$\mathbb{N}_i$	The set of the neighbor genes of gene v_i
$\mathbf{v}_i^{(s)}$	Network representation of gene v_i
$\widetilde{\mathbf{v}}_i$	Neighborhood representation of gene v_i
l	Number of hidden layers
$\mathbf{h}^{(l)}$	Output of l^{th} hidden layer
$\mathbf{W}_l$	Weight matrix for l^{th} hidden layer
$\mathbf{W}_{id}$	Weight matrix for network structure embedding
$\mathbf{W}_{out}$	Weight matrix for output layer

Table 3.1: Terms and Notations for Biological network embedding.

To model topological similarity between genes in the genetic interaction network, we define the conditional probability of gene v_j on gene v_i using a softmax function as:

$$p(v_j|v_i) = \frac{\exp(f(v_i, v_j))}{\sum_{j'=1}^{M} \exp(f(v_i, v_{j'}))} \tag{3.1}$$

which measures the likelihood of gene v_i being connected with v_j. Let function f represents the mapping of two genes v_i and v_j to their estimated proximity score. Let $p(\mathbb{N}|v)$ be the likelihood of observing a neighborhood genes $\mathbb{N}$ for a gene v. By assuming conditional independence, we can factorize the likelihood so that the likelihood of observing a neighborhood gene is independent of observing any other neighborhood gene, given a gene v_i:

$$p(\mathbb{N}_i|v_i) = \prod_{v_j \in \mathbb{N}_i} p(v_j|v_i) \tag{3.2}$$

where $\mathbb{N}_i$ represents the neighborhood genes of the gene v_i. Global structure proximity for a gene v_i can be preserved by maximizing the conditional probability over all genes in the neighborhood. Hence, we can define the likelihood function that preserve global structure proximity as:

$$\mathcal{L} = \prod_{i=1}^{M} p(\mathbb{N}_i|v_i) = \prod_{i=1}^{M} \prod_{v_j \in \mathbb{N}_i} p(v_j|v_i) \tag{3.3}$$

Let $\mathbf{v}_i^{(s)}$ denote the dense vector generated from one-hot gene ID vector, which represents topological information of that gene. Our proposed method follows direct encoding methods [15, 39] to map

genes to vector embeddings, simply known as embedding lookup:

$$\mathbf{v}_i^{(s)} = \mathbf{W}_{id} v_i \tag{3.4}$$

where $\mathbf{W}_{id} \in \mathbb{R}^{d \times M}$ is a matrix containing the embedding vectors for all genes and $v_i \in \mathbb{I}_M$ is a one-hot indicator vector indicating the column of $\mathbf{W}_{id}$ corresponding to gene v_i. The embedding matrix $\mathbf{W}_{id}$ is the trainable parameter for direct encoding methods and is directly optimized.

Learning hidden representations

Biological network embedding (BNE) captures the topological structure of the network to learn embeddings for genes. One hot representation for a gene v_i is projected to the dense vector $\mathbf{v}_i^{(s)}$ which captures the topological properties. The dense vector representation of genes are fed into a multilayer perceptron with l hidden layers. The hidden representations from each hidden layer in the model are denoted as $\mathbf{h}_i^{(0)}, \mathbf{h}_i^{(1)}, ..., \mathbf{h}_i^{(l)}$, which can be defined as :

$$\begin{aligned} \mathbf{h}_i^{(0)} &= \delta(\mathbf{W}_0 \mathbf{v}_i^{(s)} + b_0), \\ \mathbf{h}_i^{(l)} &= \delta_l(\mathbf{W}_l \mathbf{h}_i^{(l-1)} + b_l) \end{aligned} \tag{3.5}$$

where δ_l and b_l represent the activation function and the bias of the layer l respectively. $\mathbf{h}_i^{(0)}$ represents initial representation and $\mathbf{h}_i^{(l)}$ represents final representation of the input gene v_i.

At last, final representation $\mathbf{h}_i^{(l)}$ of a gene v_i from the last hidden layer is transformed to probability vector, which contains the conditional probability of all other genes to v_i:

$$\mathbf{o}_i = [p(v_1|v_i), p(v_2|v_i), \ldots, p(v_M|v_i)] \tag{3.6}$$

where $p(v_j|v_i)$ represents the probability of gene v_i being related to gene v_j and $\mathbf{o}_i$ represents the output probability vector with the conditional probability of gene v_i being connected to all other genes.

Weight matrix $\mathbf{W}_{out}$ between the last hidden layer and the output layer corresponds to the abstractive representation of neighborhood of genes. A j^{th} row from $\mathbf{W}_{out}$ refers to the compact representation of neighborhood of gene v_j, which can be denoted as $\widetilde{\mathbf{v}}_j$. The proximity score

between gene v_i and v_j can be defined as:

$$f(v_i, v_j) = \widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(l)} \tag{3.7}$$

which can be replaced into Eq. 3.1 to calculate the conditional probability:

$$p(v_j|v_i) = \frac{\exp(\widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(l)})}{\sum_{j'=1}^{M} \exp(\widetilde{\mathbf{v}}_{j'} \cdot \mathbf{h}_i^{(l)})} \tag{3.8}$$

Our model learns two latent representations $\mathbf{h}_i^{(l)}$ and $\widetilde{\mathbf{v}}_i$ for a gene v_i where $\mathbf{h}_i^{(l)}$ is the representation of gene as a node and $\widetilde{\mathbf{v}}_i$ is the representation of the gene v_i as a neighbor. Neighborhood representation $\widetilde{\mathbf{v}}_i$ can be combined with node representation $\mathbf{h}_i^{(l)}$ by addition [61, 62] to get final representation for a gene as:

$$\mathbf{z}_i = \mathbf{h}_i^{(l)} + \widetilde{\mathbf{v}}_i \tag{3.9}$$

which leads to better performance.

For an edge connecting gene v_i and v_j, we create a feature vector by combining embeddings of those genes using Hadamard product. Empirical evaluation shows features created with Hadamard product gives better performance over concatenation [15]. Then, we train a logistic classifier on these features to classify whether genes v_i and v_j interact or not.

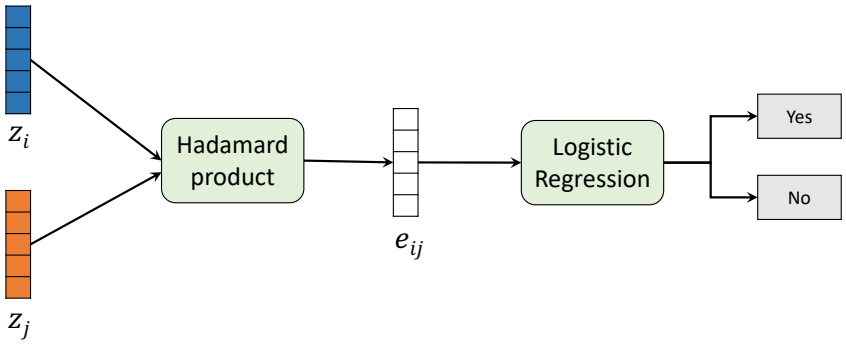


Figure 3.3: Logistic decoder for biological interaction prediction

Parameter optimization

To optimize the parameters of the proposed model, the goal is to maximize objective function mentioned in Eq. 3.8 as a function of all parameters. Let Θ be the parameters of the model that

includes $\{\mathbf{W}_{id}, \mathbf{W}_{out}, \mathbf{W}_l\}$ where $\mathbf{W}_l$ represents the weight matrices of l hidden layers. We train our model to maximize the objective function with respect to all parameters Θ :

$$\underset{\Theta}{\operatorname{argmax}} \left[\sum_{i=1}^{M} \sum_{v_j \in \mathbb{N}_i} \log \frac{\exp(\widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(l)})}{\sum_{j'=1}^{M} \exp(\widetilde{\mathbf{v}}_{j'} \cdot \mathbf{h}_i^{(l)})} \right] \tag{3.10}$$

Maximizing this objective function with respect to Θ is computationally expensive, which requires the calculation of partition function $\sum_{j'=1}^{M} \exp(\widetilde{\mathbf{v}}_{j'} \cdot \mathbf{h}_i^{(l)})$ for each gene. To calculate a single probability, we need to aggregate all genes in the network. To address this problem, we adopt the approach of negative sampling [39] which samples negative interactions, interactions with no evidence of their existence, according to some noise distribution for each edge e_{ij}. This approach allows us to sample a small subset of genes from the network as negative samples for a gene, considering that the genes on the selected subset don't fall in the neighborhood $\mathbb{N}_i$ of the gene. Above objective function enhances the similarity of a gene v_i with its neighborhood genes $v_j \in \mathbb{N}_i$ and weakens the similarity with genes not in its neighborhood genes $v_j \notin \mathbb{N}_i$. It is inappropriate to assume that the two genes in the network are not related if they are not connected. It may be the case that there is not enough experimental evidence to support that they are related yet. Thus, forcing the dissimilarity of a gene with all other genes, not in its neighborhood $\mathbb{N}_i$ seems to be inappropriate.

We adopt Adaptive Moment Estimation (Adam) optimization [63], which is an extension to stochastic gradient descent, to optimize Eq. 3.10. To address the overfitting problem, regularization like dropout [64] and batch normalization [65] is added to hidden layers.

3.3 Experimental setup

3.3.1 Datasets

We evaluate our model using two interaction datasets from the BioGRID database (2017 version 3.4.153) [66] to evaluate the predictive performance of our model. Self-interactions and redundant interactions are removed from interaction datasets. The statistics of the datasets are shown in Table 3.2.

For the experimental evaluation, we split the interaction dataset into training, validation, and test set. The best set of hyperparameters are selected based on the model's performance on the

Datasets	No. of Genes	No. of Interactions
Yeast	5,950	544,652
E. coli	4,511	148,340

Table 3.2: Statistics of the interaction datasets from the BioGRID database.

validation dataset. The interaction dataset only contains the positive interactions i.e. interactions with pieces of evidence supporting their existence. By adopting negative sampling, we can train the model only by taking positive interactions in the training set. However, negative interactions are also crucial to evaluate the model on the validation and test set. To this aim, we randomly sample an equal number of negative interactions i.e. non-edges in the network.

3.3.2 Baselines

We compare our method with three baselines for network embedding: Isomap, LINE, node2vec. These baselines are also designed to capture the topological properties of the nodes in the network and have demonstrated good performance on the link prediction tasks. In particular, Isomap [16] computes all-pairs shortest distances to create a distance matrix and performs singular value decomposition (SVD) to learn latent representations of the nodes. In contrast, LINE [18] preserves the first-order and second-order proximity of the network separately to learn the embeddings. On the other hand, node2vec [15] performs truncated biased random walks to generate a sequence of co-occurring nodes in the network. Then, the representation for nodes is learned by applying the Skip-gram model on the set of generated node sequences.

3.3.3 Hyperparameter selection

We implement the proposed model in TensorFlow [67]. We construct a neural network with single hidden layer ($l = 1$). The set of hyperparameters and their possible values consider in this work are provided in Table 3.3.

The best set of hyperparameters are selected based on empirical evaluation and Table 3.4 summarizes the optimal parameters tuned on validation datasets.

Hyperparameters	Values
Batch size	8, 16, 32, 64, 128, 256
Learning rate	0.1, 0.01, 0.005, 0.002, 0.001, 0.0001
Negative samples	2, 5, 10, 15, 20
Embedding dimension d	32, 64, 128, 256

Table 3.3: Hyperparameters and the set of considered values

Hyperparameter	Yeast	Ecoli
Batch size	256	128
Learning rate	0.005	0.002
Negative samples	10	10
Embedding dimension (d)	128	128

Table 3.4: Optimal hyperparamter settings of the model

3.4 Results and discussion

In this section, we evaluate and compare various embedding methods for biological interaction prediction task. We show the effect of network sparsity on the performance of the proposed model.

3.4.1 Gene interaction prediction

For this experiment, we randomly remove 50% of the interactions from the network as a test set and train the model with the remaining interactions. Then, the trained model is evaluated on a test set to see how well the model can predict these missing interactions. Table 3.5 shows that the proposed model outperforms other baselines by a huge margin. In particular, our method gains 0.287 in AUROC, 0.196 in AP for yeast, and 0.371 in AUROC, 0.259 in AP for E-coli over Isomap. Similarly, our method achieves significant improvement over node2vec.

3.4.2 Robustness to network sparsity

We further investigate the robustness of the proposed model to network sparsity. To this aim, we create a testing set with 10% of interactions and create training sets with different percentage of interactions. Then, we train the model on these training sets and compare their performances on the test set. Figure 3.4 shows that the performance of the model increases with the increase in

Datasets	Yeast		E-coli	
	AUROC	**AUPR**	**AUROC**	**AUPR**
Isomap [68]	0.507	0.588	0.559	0.672
LINE [18]	0.726	0.686	0.897	0.851
node2vec [15]	0.739	0.708	0.912	0.862
BioNetEmbedding	**0.787**	**0.784**	**0.930**	**0.931**

Table 3.5: Area under ROC curve (AUROC) and Area under PR curve (AUPR) for gene interaction prediction.

training interactions. Specifically, for the network with less portion of training interactions (< 0.5), the performance improves by a huge margin. However, the performance remains stable when the percentage of training interactions is greater than 50%.

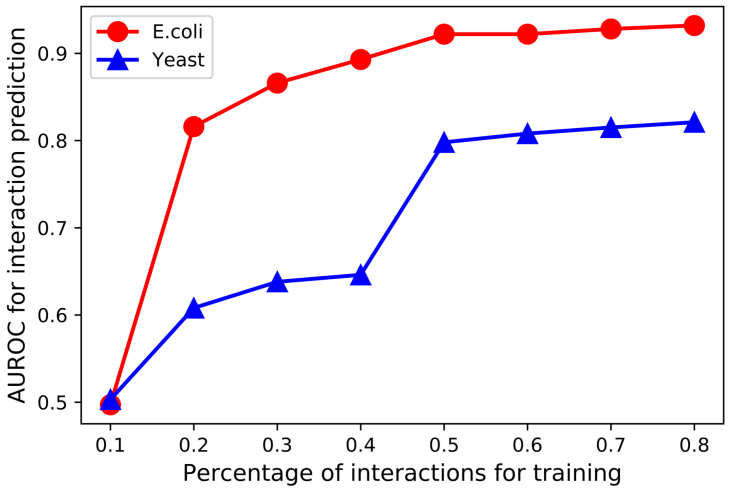


Figure 3.4: AUROC comparison of model's performance with respect to network sparsity.

3.5 Application to biomedical networks

While biological system represents the study of living organisms and how they relate to the environment, biomedical science is specific to human biology and involves the study of interconnections between human molecular entities and various diseases. With recent advancements of high-throughput technologies, a large number of heterogeneous datasets have been generated, informing about different aspects of human biological systems. As discussed before, the network representation of

the biological system simplifies the computational modeling and analysis of these datasets. For example, various diseases and the set of drugs used to treat these diseases can be represented as a network and the task of link prediction, in this case, is to predict the set of novel drugs used to treat a particular disease. In particular, network-based approaches have been explored to find a set of potential drugs for COVID-19 diseases [69, 70].

For this experiment, we explore if our proposed model for gene interaction prediction can be applied for link prediction tasks in other biomedical networks. To this aim, we consider four datasets to evaluate the link prediction performance (Table 3.6):

- **Comparative Toxicogenomics Database Drug-Disease Association (DDA)**: Comparative Toxicogenomics Database (CTD) [71] provides information about chemical-diseases associations. We extract the curated associations from CTD and create a network with 92,813 edges between 12,765 nodes. We term this dataset as "CTD DDA".
- **National Drug File Reference Terminology Drug-Disease Association**: Another drug-disease association (DDA) network is constructed from the National Drug File Reference Terminology (NDF-RT) in UMLS [72]. The association between drugs and diseases involves *may treat* and *may be treated by* relationships in NDF-RT. The constructed DDA network has 13, 545 nodes with 56,515 edges. We term this dataset as "NDFRT DDA".
- **Drug-Drug interactions (DDI)**: Drugbank [73] is a freely accessible online database that stores detailed information about different drugs and their associations with other drugs. We construct a DDI network where nodes represent drugs and the edges represent the side effects of two drugs if taken simultaneously. The DDI network has 2191 drugs as nodes and 242,027 associations between them. We term this network as "Drugbank DDI".
- **Protein-protein interaction (PPI)**: Human protein-protein interaction network is constructed by extracting PPIs from STRING database [74]. STRING database also provides the confidence score for each interaction indicating the possibility of that interaction to be a true positive interaction. We selected 359,776 interactions between 15,131 proteins with a confidence score larger than 0.7. We term this network as "STRING PPI".

Given a biomedical network $\mathcal{G}$, we aim to predict missing interactions in the network. For this experiment, we split the biomedical interactions as training, and test set in an 8:2 ratio. We compare the performance of our method with five matrix factorization-based methods, three random walk-based methods, and three neural network-based methods discussed in [24]. Table 3.7 shows that

Dataset	# Nodes	# Edges
CTD DDA	9,580 drugs, 3,185 diseases	92,813
NDFRT DDA	12,337 drugs, 1,208 diseases	56,515
Drugbank DDI	2,191 drugs	242,027
STRING PPI	15,131 proteins	359,776

Table 3.6: Summary of the datasets used in our experiments.

our proposed method achieves significant improvement over several baselines. GraRep achieves the best performance among the baselines on CTD DDA, NDFRT DDA, DrugBank DDI, and struc2vec achieves the best performance on the STRING PPI dataset. Moreover, our method achieves 0.93% gain in CTD DDA dataset, 1.46% in NDFRT DDA, 3.89% in Drugbank DDI over GraRep, 7.92% over struc2vec in STRING PPI.

Method		CTD DDA	NDFRT DDA	Drugbank DDI	STRING PPI
MF	Laplacian	0.856±0.004	0.930±0.003	0.796±0.002	0.639±0.021
	SVD	0.936 ± 0.002	0.779±0.003	0.919±0.001	0.867±0.001
	GF	0.884 ± 0.004	0.720±0.006	0.882±0.003	0.817±0.005
	HOPE	0.951 ± 0.001	0.949±0.001	0.923±0.001	0.839±0.001
	GraRep	0.960 ± 0.001	0.963±0.001	0.925±0.001	0.894±0.001
RW	DeepWalk	0.929 ± 0.002	0.783±0.004	0.921±0.001	0.884±0.001
	node2vec	0.911 ± 0.002	0.819±0.005	0.902±0.001	0.828±0.003
	struc2vec	0.965 ± 0.001	0.958±0.001	0.904±0.001	0.909±0.001
NN	LINE	0.965 ± 0.001	0.962±0.002	0.905±0.002	0.859±0.003
	SDNE	0.935 ± 0.010	0.944±0.004	0.911±0.006	0.884±0.008
	GAE	0.937 ± 0.001	0.813±0.007	0.917±0.001	0.900±0.001
Ours		**0.974 ± 0.005**	**0.977±0.002**	**0.961±0.003**	**0.981±0.001**

Table 3.7: Average AUC score (with one standard deviation) averaged over five independent runs for link prediction.

This demonstrates that our proposed model can be applied for various biomedical link prediction tasks to improve the prediction performance.

3.6 Conclusion

We introduced a neural network model for gene interaction prediction using biological network data. Our model, termed BioNetEmbedding, learns the representation for genes by capturing the first-order and second-order network proximity from interaction networks. Experiments on real-world datasets suggest that our proposed model is capable of encoding network properties for interaction prediction. Furthermore, the results indicate that our model outperforms several previously proposed methods by a huge margin. Moreover, our method is applicable to other biomedical networks that have network structure only and achieves superior performance compared to various baselines. In following chapters, we will discuss our novel approach to integrate heterogeneous datasets with network structure.

Chapter 4

Representation Learning of biological networks using local network properties and continuous node features

Previously in Chapter 3, we explored the problem of interaction prediction using a biological network constructed from known interactions. Since the constructed interaction network is noisy, sparse, and incomplete, this limits the model's capability to predict novel interaction for genes with zero or few interactions. To solve this problem, we propose to integrate additional information for entities with network properties. In particular, gene expression data has been well studied in literature to uncover gene regulatory interactions, based on the mutual dependencies between the expression of regulatory genes and their target genes. We thus hypothesize that the integration of expression data allows the models to identify the pair of genes that are co-expressed. This chapter describes the novel neural network that integrates gene expression data with gene interaction networks to predict novel interactions. In the next chapter, we will discuss the integration of sequential features (e.g. protein sequences) with interaction networks.

4.1 Introduction

A comprehensive study of gene interactions (GIs) provides means to identify the functional relationship between genes and their corresponding products, as well as insights into underlying biological phenomena that are critical to understanding phenotypes in health and disease conditions [8,9,10]. Since advancements in measurement technologies have led to numerous high-throughput datasets, there is great value in developing efficient computational methods capable of automatically extracting and aggregating meaningful information from heterogeneous datasets to infer gene interactions.

Although a wide variety of machine learning models have been developed to analyze high-throughput datasets for GI prediction [75], there are still some major challenges, such as efficient analysis of large heterogeneous datasets, integration of biological information, and effective feature engineering. To address these challenges, we propose a novel deep learning framework to integrate diverse biological information for GI network inference [76].

Our proposed method frames GI network inference as a problem of network embedding. In particular, we represent gene interactions as a network of genes and their interactions and create a deep learning framework to automatically learn an informative representation that integrates both the topological property and the gene expression property. A key insight behind our gene network embedding method is the "guilt by association" assumption [77], that is, genes that are co-localized or have similar topological roles in the interaction network are likely to be functionally correlated. This insight not only allows us to discover similar genes and proteins but also to infer the properties of unknown ones. Our network embedding generates a lower-dimensional vector representation of the gene topological characteristics. The relationships between genes including higher-order topological properties are captured by the distances between genes in the embedding space. The new low-dimensional representation of a GI network can be used for various downstream tasks, such as gene function prediction, gene interaction prediction, and gene ontology reconstruction [60].

Furthermore, since the network embedding method can only preserve the topological properties of a GI network (Chapter 3), and fails to generalize for genes with no interaction information, our scalable deep learning method also integrates heterogeneous gene information, such as expression data from high throughput technologies, into the GI network inference. Our method projects genes with similar attributes closer to each other in the embedding space, even if they may not have similar topological properties. The results show that by integrating additional gene information in the network embedding process, the prediction performance is improved significantly.

In summary, our contributions are as follows:

- We propose a novel deep learning framework to learn lower-dimensional representations while preserving topological and attribute proximity of GI networks.
- We evaluate the prediction performance on the datasets of two organisms based on the embedded representation and achieve significantly better predictions than the strong baselines.
- Our method can predict new gene interactions that are validated on an up-to-date GI database.

4.2 Methods

4.2.1 Preliminaries

We formally define the problem of gene network inference as a network embedding problem using the concepts of topological and attribute proximity as demonstrated in Figure 4.1.

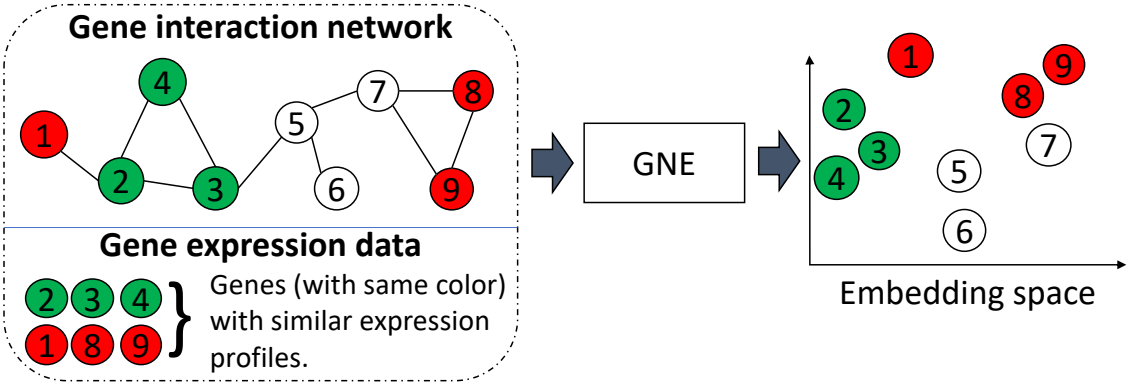


Figure 4.1: An illustration of gene network embedding (GNE). GNE integrates gene interaction network and gene expression data to learn a lower-dimensional representation. The nodes represent genes, and the genes with the same color have similar expression profiles. GNE groups genes with similar network topology, which are connected or have a similar neighborhood in the graph, and attribute similarity (similar expression profiles) in the embedded space.

According to Definition 2.1, $\mathcal{V}$ denotes genes or proteins, $\mathcal{E}$ denotes interactions between genes/proteins and X denotes gene expression data.

Genes directly connected with a gene v_i in the gene network denote the local network structure of gene v_i. We define local network structures as the first-order proximity of a gene.

First-order and second-order proximity have been defined in Definition 3.1 and 3.2. Integrating first-order and second-order proximities simultaneously can help to preserve the topological properties

of the gene network. To generate a more comprehensive representation of the genes, it is crucial to integrate gene expression data as gene attributes with their topological properties. Besides preserving topological properties, gene expression provides additional information to predict the network structure.

Definition 4.1. *(**Attribute proximity**): Attribute proximity denotes the similarity between the expression of genes. In this chapter, we consider the similarity in expression profiles of genes as the attribute proximity.*

We propose to map gene network structure and their attributes to a low dimensional space $\mathbf{z} \in \mathbb{R}^d$. We thus investigate both topological and attribute proximity for gene network embedding, which is defined as follows:

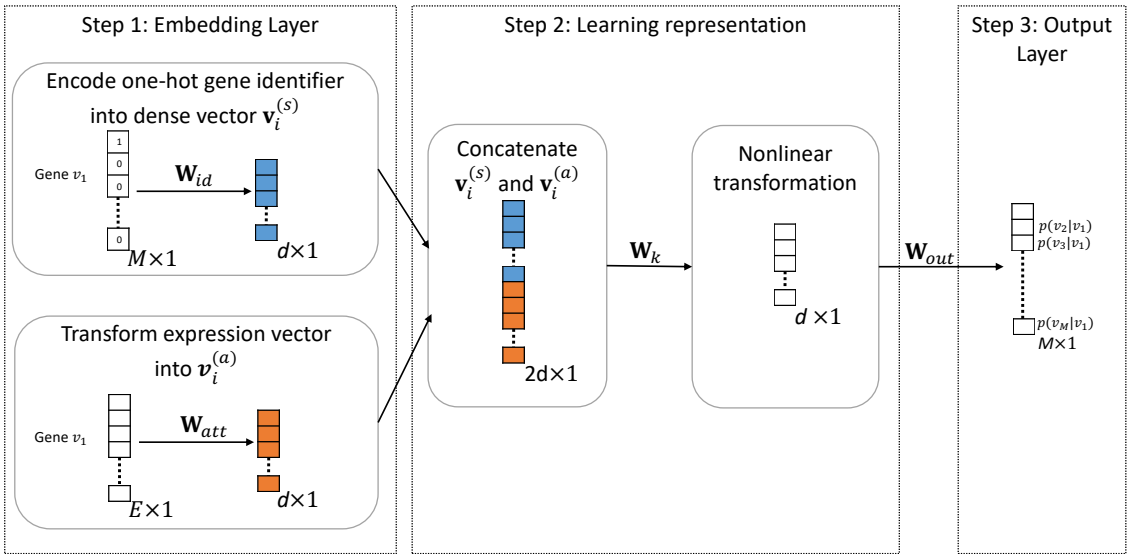


Figure 4.2: Overview of Gene Network Embedding (GNE) Framework for gene interaction prediction.

4.2.2 Gene network embedding (GNE) model

Our deep learning framework jointly utilizes gene network structure and gene expression data to learn a unified representation for the genes as shown in Figure 4.2. The representations capture the complex statistical relationships between the gene network structure and gene expression data. We summarize GNE model as:

- One-hot encoded representation of gene is encoded to dense vector $\mathbf{v}_i^{(s)}$ of dimension $d \times 1$

which captures topological properties and expression vector of gene is transformed to $\mathbf{v}_i^{(a)}$ of dimension $d \times 1$ which aggregates the attribute information.

- Next, concatenation of two embedded vectors (creates vector with dimension $2d \times 1$) allows to combine strength of both network structure and attribute modeling. Then, nonlinear transformation of concatenated vector enables GNE to capture complex statistical relationships between network structure and attribute information and learn better representations.
- Finally, these learned representation of dimension $d \times 1$ is transformed into a probability vector of length $M \times 1$ in output layer, which contains the predictive probability of gene v_i to all the genes in the network. Conditional probability $p(v_j|v_i)$ on output layer indicates the likelihood that gene v_j is connected with gene v_i.

In addition to notations listed in Table 3.1, we list additional variables for this framework in Table 4.1.

Symbol	Definitions
E	Number of experiments used to measure expression levels of genes
$\mathbf{v}_i^{(a)}$	Attribute representation of gene v_i
$\mathbf{v}_i$	Concatenated representation of network properties and expression data
$\mathbf{W}_{att}$	Weight matrix for attribute embedding

Table 4.1: Terms and Notations for Gene network embedding

Next, we discuss different components of GNE in detail.

Gene network structure modeling

GNE framework preserves first-order and second-order proximity of genes in the gene network as discussed in Section 3.2.

Gene expression modeling

In addtion to encoding network structure, GNE encodes the expression data from microarray experiments to the dense representation using a non-linear transformation. The amount of mRNA

produced during transcription measured over a number of experiments helps to identify similarly expressed genes. Since expression data have inherent noise [78], transforming expression data using a non-linear transformation can be helpful to uncover the underlying representation. Let $\mathbf{x}_i$ be the vector of expression values of gene v_i measured over E experiments. Using non-linear transformation, we can capture the non-linearities of expression data of gene v_i as:

$$\mathbf{v}_i^{(a)} = \delta_a(\mathbf{W}_{att} \cdot \mathbf{x}_i) \tag{4.1}$$

where $\mathbf{v}_i^{(a)}$ represents the lower dimensional attribute representation vector for gene v_i. $\mathbf{W}_{att}$, and δ_a represents the weight matrix, and activation function of attribute transformation layer respectively.

We use deep model to approximate the attribute proximity by capturing complex statistical relationships between attributes and introducing non-linearities, similar to structural embedding.

GNE integration

GNE models the integration of network structure and attribute information to learn more comprehensive embeddings for gene networks. GNE takes two inputs: one for topological information of a gene as one hot gene ID vector and another for its expression as an attribute vector. Each input is encoded to its respective embeddings. One hot representation for a gene v_i is projected to the dense vector $\mathbf{v}_i^{(s)}$ which captures the topological properties. Non-linear transformation of attribute vector generates compact representation vector $\mathbf{v}_i^{(a)}$. Previous work [18] combines heterogeneous information using the late fusion approach. However, the late fusion approach is the approach of learning separate models for heterogeneous information and integrating the representations learned from separate models. On the other hand, the early fusion combines heterogeneous information and train the model on combined representations [79]. We thus propose to use the early fusion approach to combine them by concatenating. As a result, learning from topological and attribute information can complement each other, allowing the model to learn their complex statistical relationships as well. Embeddings from topological and attribute information are concatenated into a vector as:

$$\mathbf{v}_i = [\mathbf{v}_i^{(s)} \quad \lambda\mathbf{v}_i^{(a)}] \tag{4.2}$$

where λ is the importance of gene expression information relative to topological information.

The concatenated vectors are fed into a multilayer perceptron with l hidden layers. The hidden representations from each hidden layer in GNE are denoted as $\mathbf{h}_i^{(0)}, \mathbf{h}_i^{(1)}, ..., \mathbf{h}_i^{(l)}$, which can be

defined as :

$$\begin{aligned} \mathbf{h}_i^{(0)} &= \delta(\mathbf{W}_0 \mathbf{v}_i + b^{(0)}), \\ \mathbf{h}_i^{(l)} &= \delta_l(\mathbf{W}_l \mathbf{h}_i^{(l-1)} + b^{(l)}) \end{aligned} \tag{4.3}$$

where δ_l represents the activation function of layer l. $\mathbf{h}_i^{(0)}$ represents initial representation and $\mathbf{h}_i^{(l)}$ represents final representation of the input gene v_i. Transformation of input data using multiple non-linear layers has shown to improve the representation of input data [80]. Moreover, stacking multiple layers of non-linear transformations can help to learn high-order statistical relationships between topological properties and attributes.

At last, final representation $\mathbf{h}_i^{(l)}$ of a gene v_i from the last hidden layer is transformed to probability vector, which contains the conditional probability of all other genes to v_i:

$$\mathbf{o}_i = [p(v_1|v_i), p(v_2|v_i), \ldots, p(v_M|v_i)] \tag{4.4}$$

where $p(v_j|v_i)$ represents the probability of gene v_i being related to gene v_j and $\mathbf{o}_i$ represents the output probability vector with the conditional probability of gene v_i being connected to all other genes.

Weight matrix $\mathbf{W}_{out}$ between the last hidden layer and the output layer corresponds to the abstractive representation of neighborhood of genes. A j^{th} row from $\mathbf{W}_{out}$ refers to the compact representation of neighborhood of gene v_j, which can be denoted as $\widetilde{\mathbf{v}}_j$. The proximity score between gene v_i and v_j can be defined as:

$$f(v_i, v_j) = \widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(l)} \tag{4.5}$$

which can be replaced into Eq. 3.1 to calculate the conditional probability:

$$p(v_j|v_i) = \frac{\exp(\widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(l)})}{\sum_{j'=1}^{M} \exp(\widetilde{\mathbf{v}}_{j'} \cdot \mathbf{h}_i^{(l)})} \tag{4.6}$$

Our model learns two latent representations $\mathbf{h}_i^{(l)}$ and $\widetilde{\mathbf{v}}_i$ for a gene v_i where $\mathbf{h}_i^{(l)}$ is the representation of gene as a node and $\widetilde{\mathbf{v}}_i$ is the representation of the gene v_i as a neighbor. Neighborhood representation $\widetilde{\mathbf{v}}_i$ can be combined with node representation $\mathbf{h}_i^{(l)}$ by addition [61, 62] to get final representation for a gene as:

$$\mathbf{z}_i = \mathbf{h}_i^{(l)} + \widetilde{\mathbf{v}}_i \tag{4.7}$$

which returns us better performance results.

For an edge connecting gene v_i and v_j, we create a feature vector by combining embeddings of those genes using Hadamard product. Empirical evaluation shows features created with Hadamard product gives better performance over concatenation [15]. Then, we train a logistic classifier on these features to classify whether genes v_i and v_j interact or not. We follow same optimization for the parameters of GNE model as discussed in Section 3.2.

4.2.3 Experimental setup

We evaluate our model using two real organism datasets. We take gene interaction network data from the BioGRID database [66] and gene expression data from DREAM5 challenge [81]. We use two interaction datasets from the BioGRID database (2017 released version 3.4.153 and 2018 released version 3.4.158) to evaluate the predictive performance of our model. Self-interactions and redundant interactions are removed from interaction datasets. The statistics of the datasets are shown in Table 4.2.

		#(Interactions)		Expression data
Datasets	#(Genes)	2017 version	2018 version	#(Experiments)
Yeast	5,950	544,652	557,487	536
E. coli	4,511	148,340	159,523	805

Table 4.2: Statistics of the interaction datasets from BioGRID and the gene expression data from DREAM5 challenge.

We evaluate the learned embeddings to infer gene network structure. We randomly hold out a fraction of interactions as the validation set for hyper-parameter tuning. Then, we divide the remaining interactions randomly into training and testing datasets with an equal number of interactions. Since the validation set and the test set contains only positive interactions, we randomly sample an equal number of gene pairs from the network, considering the missing edge between the gene pairs represents the absence of interactions. Given the gene network G with a fraction of missing interactions, the task is to predict these missing interactions.

We compare the GNE model with five competing methods. Correlation directly predicts the in-

teractions between genes based on the correlation of expression profiles. Then, the following three baselines (Isomap, LINE, and node2vec) are network embedding methods. Specifically, node2vec is a strong baseline for structural network embedding. We evaluate the performance of GNE against the following methods:

- **Correlation** [82]: It computes Pearson's correlation coefficient between all genes and the interactions are ranked via correlation scores, i.e., highly correlated gene pairs receive higher confidence.
- **Isomap** [68]: It computes all-pairs shortest-path distances to create a distance matrix and performs singular-value decomposition of that matrix to learn a lower-dimensional representation. Genes separated by the distance less than threshold ϵ in embedding space are considered to have the connection with each other and the reliability index, a likelihood indicating the interaction between two genes, is computed using FSWeight [83].
- **LINE** [18]: Two separate embeddings are learned by preserving first-order and second-order proximity of the network structure respectively. Then, these embeddings are concatenated to get final representations for each node.
- **node2vec** [15]: It learns the embeddings of the node by applying the Skip-gram model to node sequences generated by a biased random walk. We tuned two hyper-parameters p and q that control the random walk.

Note that competing methods such as Isomap, LINE, and node2vec are designed to capture only the topological properties of the network. For the fair comparison with GNE that additionally integrates expression data, we concatenate attribute feature vector with learned gene representation to extend baselines by including the gene expression. We name these variants as Isomap+, LINE+, and node2vec+.

We have implemented GNE with the TensorFlow framework [67]. The parameter settings for GNE are determined by their performance on the validation set. We randomly initialize GNE's parameters, optimizing with mini-batch Adam. We test the batch size of [8, 16, 32, 64, 128, 256] and learning rate of [0.1, 0.01, 0.005, 0.002, 0.001, 0.0001]. We test the number of negative samples to be [2, 5, 10, 15, 20] as suggested by [39]. We test the embedding dimension d of [32, 64, 128, 256] for all methods. Also, we evaluate model's performance with respect to different values of λ [0, 0.2, 0.4, 0.6, 0.8, 1], which is discussed in more detail later. The parameters are selected based

Hyperparameter	Yeast	Ecoli
Batch size	256	128
Learning rate	0.005	0.002
Negative samples	10	10
Embedding dimension (d)	128	128

Table 4.3: Optimal parameter settings for GNE model

on empirical evaluation and Table 4.3 summarizes the optimal parameters tuned on validation data sets.

To capture the non-linearity of gene expression data, we choose Exponential Linear Unit (ELU) [84] activation function, which corresponds to δ_a in Eq. 4.1. Also, ELU activation avoids vanishing gradient problem and provides improved learning characteristics in comparison to other methods. We use a single hidden layer ($l = 1$) with hyperbolic tangent activation (Tanh) to model complex statistical relationships between topological properties and attributes of the gene. The choice of ELU for attribute transformation and Tanh for the hidden layer shows better performance upon empirical evaluation.

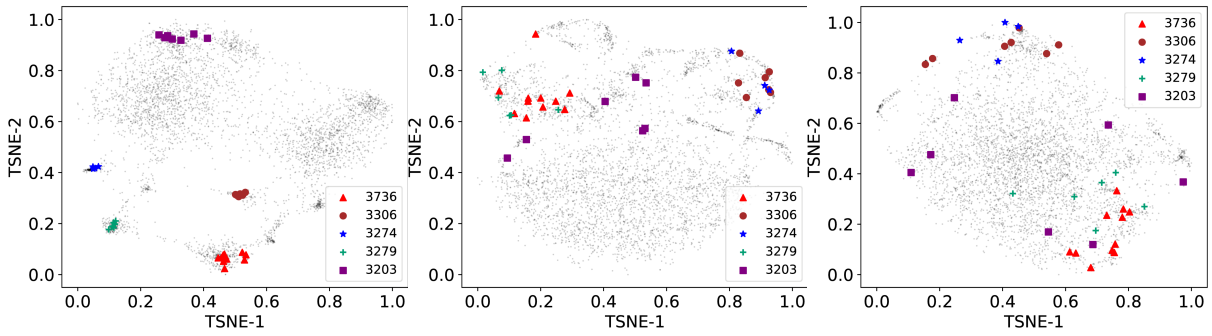
We use the area under the ROC curve (AUROC) and area under the precision-recall curve (AUPR) [85] to evaluate the rankings generated by the model for interactions in the test set. These metrics are widely used in evaluating the ranked list of predictions in gene interaction [75].

4.3 Results and discussion

We evaluate the ability of our GNE model to predict gene interaction of two real organisms. We present empirical results of our proposed method against other methods.

4.3.1 Analysis of gene embeddings

We visualize the embedding vectors of genes learned by GNE. We take the learned embeddings, which specifically model the interactions by preserving topological and attribute similarity. We embed these embeddings into a 2D space using t-SNE package [86] and visualize them (Figure 4.3). For comparison, we also visualize the embeddings learned by structure-preserving deep learning methods, such as LINE, and node2vec.



(a) (b) (c)

Figure 4.3: Visualization of learned embeddings for genes on E. coli. Genes are mapped to the 2D space using the t-SNE package with learned gene representations ($\mathbf{z}_i, i = 1, 2, \ldots, M$) from different methods: (a) GNE, (b) LINE, and (C) node2vec as input. Operons 3203, 3274, 3279, 3306, and 3736 of E. coli are visualized and show clustering patterns. Best viewed on screen.

In E. coli, a substantial fraction of functionally related genes is organized into operons, which are the group of genes that interact with each other and are co-regulated [87]. Since this concept fits well with the topological and attributes proximity implemented in GNE, we expect GNE to place genes within an operon close to each other in the embedding space. To evaluate this, we collect information about operons of E. coli from the DOOR database and visualize the embeddings of genes within these operons (Figure 4.3).

Figure 4.3 reveals the clustering structure that corresponds to the operons on E. coli. For example, operon with operon id 3306 consists of seven genes: rsxA, rsxB, rsx, rsxD, rsxG, rsxE, and nth that are involved in electron transport. GNE infers similar representations for these genes, resulting in localized projection in the 2D space. Similarly, other operons also show similar patterns.

To test if the pattern in Figure 4.3 holds across all operons, we compute the average Euclidean distance between each gene's vector representation and vector representations of other genes within the same operon. Genes within the same operon have significantly similar vector representation $\mathbf{z}_i$ than expected by chance (p-value $= 1.75e - 127$, 2-sample KS test).

Thus, the analysis here indicates that GNE can learn similar representations for genes with similar topological properties and expression.

4.3.2 Gene interaction prediction

We randomly remove 50% of interactions from the network and compare various methods to evaluate their predictions for 50% missing interactions. Table 4.4 shows the performance of GNE and other methods on gene interaction prediction across different datasets. As our method significantly outperforms other competing methods, it indicates the informativeness of gene expression in predicting missing interactions. Also, our model is capable of integrating attributes with topological properties to learn better representations.

Methods	Yeast		E. coli	
	AUROC	AUPR	AUROC	AUPR
Correlation	0.582	0.579	0.537	0.557
Isomap	0.507	0.588	0.559	0.672
LINE	0.726	0.686	0.897	0.851
node2vec	0.739	0.708	0.912	0.862
Isomap+	0.653	0.652	0.644	0.649
LINE+	0.745	0.713	0.899	0.856
node2vec+	0.751	0.716	0.871	0.826
GNE (Topology)	0.787	0.784	0.930	0.931
GNE (our model)	**0.825***	**0.821***	**0.940***	**0.939***

Table 4.4: Area under ROC curve (AUROC) and Area under PR curve (AUPR) for gene Interaction Prediction. + indicates the concatenation of expression data with learned embeddings to create final representation. * denotes that GNE significantly outperforms node2vec at 0.01 level paired t-test. Note that method that achieves the best performance is bold faced.

We compare our model with a correlation-based method, that takes only expression data into account. Our model shows significant improvement of 0.243 (AUROC), 0.242 (AUPR) on yeast and 0.403 (AUROC), 0.382 (AUPR) on E. coli over correlation-based methods. This improvement suggests the significance of the topological properties of the gene network.

The network embedding method, Isomap, performs poorly in comparison to correlation-based methods on yeast because of its limitation on network inference. Deep learning-based network embedding methods such as LINE and node2vec show a significant gain over Isomap and correlation-based methods. node2vec outperforms LINE across two datasets. Moreover, GNE trained only with topological properties outperforms these structured-based deep learning methods (Table 4.4). However, these methods don't consider the attributes of the gene that we suggest to contain useful information for gene interaction prediction. By adding expression data with topological properties, GNE outperforms structure-preserving deep embedding methods across both datasets.

Focusing on the results corresponding to the integration of expression data with topological properties, we find that the method of integrating the expression data plays an essential role in the performance. The performance of node2vec+ (LINE+, Isomap+) shows little improvement with the integration of expression data on yeast. However, node2vec+ (LINE+, Isomap+) has no improvement or decline in performance on E. coli. The decline in performance indicates that merely concatenating the expression vector with learned representations for the gene is insufficient to capture the rich information in expression data. The late fusion approach of combining the embedding vector corresponding to the topological properties of the gene network and the feature vector representing expression data has no significant improvement in the performance (except Isomap). In contrast, our model incorporates gene expression data with topological properties by the early fusion method and shows significant improvement over other methods.

4.3.3 Robustness to network sparsity

We investigate the robustness of our model to network sparsity. We hold out 10% interactions as the test set and change the sparsity of the remaining network by randomly removing a portion of remaining interactions. Then, we train GNE to predict interactions in the test set and evaluate the change in performance to network sparsity. We evaluate two versions of our implementations: GNE with only topological properties and GNE with topological properties and expression data. The result is shown in Figure 4.4.

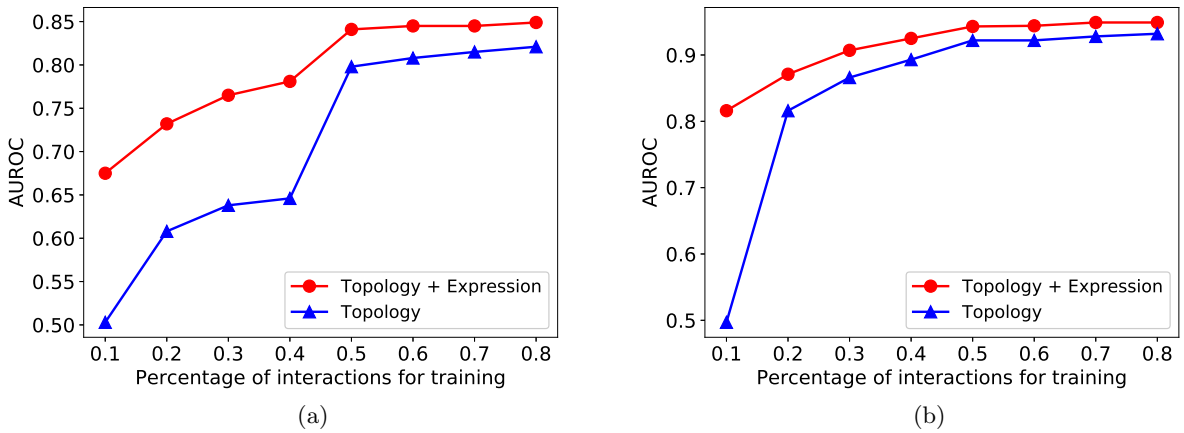


Figure 4.4: AUROC comparison of GNE's performance with respect to network sparsity. (a) yeast (b) E. coli. Integration of expression data with topological properties of the gene network improves the performance for both datasets.

Figure 4.4 shows that our method's performance improves with an increase in the number of training interactions across datasets. Also, our method's performance improves when expression data is integrated with the topological structure. Specifically, GNE trained on 10% of total interactions and attributes of yeast shows a significant gain of 0.172 AUROC (from 0.503 to 0.675) over GNE trained only with 10% of total interactions. Similarly, GNE improves the AUROC from 0.497 to 0.816 for E. coli with the same setup as shown in Figure 4.4. The integration of gene expression data results in less improvement when we train GNE on a relatively large number of interactions.

Moreover, the performance of GNE trained with 50% of total interactions and expression data is comparable to be trained with 80% of total interactions without gene expression data as shown in Figure 4.4. The integration of expression data with topological properties into the GNE model has more improvement on E. coli than yeast when we train with 10% of total interactions for each dataset. The reason for this is likely the difference in the number of available interactions for yeast and E. coli (Table 4.2). This indicates the informativeness of gene expression when we have few interactions and supports the idea that the integration of expression data with topological properties improves gene interaction prediction.

4.3.4 Hyperparameter validation

GNE involves the parameter λ that controls the importance of gene expression information relative to topological properties of gene network as shown in Eq. 4.2. We examine how the choice of the parameter λ affects our method's performance. Figure 4.5 shows the comparison of our method's performance with different values of λ when GNE is trained on the varying percentage of total interactions.

We evaluate the impact of λ on range [0, 0.2, 0.4, 0.6, 0.8, 1]. When λ becomes 0, the learned representations model only topological properties. In contrast, setting the high value for λ makes GNE learn only from attributes and degrades its performance. Therefore, our model performs well when λ is within [0, 1].

Figure 4.5 shows that the integration of expression data improves the performance of GNE to predict gene interactions. The impact of λ depends on the number of interactions used to train GNE. If GNE is trained with few interactions, integration of expression data with topological properties plays a vital role in predicting missing interactions. As the number of training interactions increases, integration of expression data has less impact but still improves the performance over only topological properties.

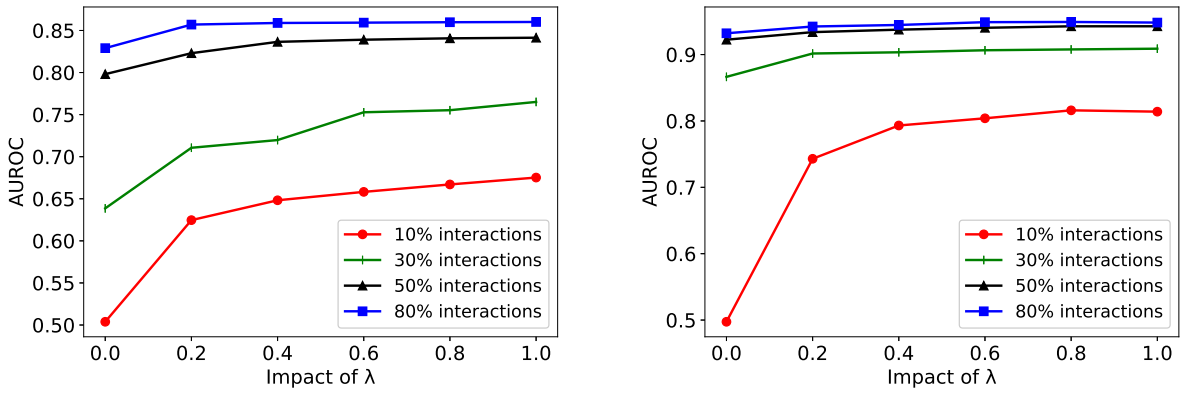


Figure 4.5: Impact of λ on GNE's performance trained with different percentages of interactions. (a) yeast (b) E. coli. Different lines indicate performance of GNE trained with different percentages of interactions.

Figure 4.4 and 4.5 demonstrate that the expression data contributes to the increase in AUROC by nearly 0.14 when interactions are less than 40% for yeast and about 0.32 when interactions are less than 10% for E. coli. More topological properties and attributes are required for yeast than E. coli. It may be related to the fact that yeast is a more complex species than E. coli. Moreover, we can speculate that more topological properties and attributes are required for higher eukaryotes like humans. In humans, GNE that integrates topological properties with attributes may be more successful than the methods that only use either topological properties or attributes.

This demonstrates the sensitivity of GNE to parameter λ. This parameter λ has a considerable impact on our method's performance and should be appropriately selected.

4.3.5 Investigation of GNE's predictions

We investigate the predictive ability of our model in identifying new gene interactions. For this aim, we consider two versions of BioGRID interaction datasets at two different time points (2017 and 2018 version), where the older version is used for training, and the newer one is used for testing the model (temporal holdout validation). The 2018 version contains 12,835 new interactions for yeast and 11,185 new interactions for E. coli than the 2017 version. GNE's performance trained with 50% and 80% of total interactions is comparable for both yeast and E. coli (Figures 4 and 5). We thus train our model with 50% of total interactions from the 2017 version to learn the embeddings for

genes and demonstrate the impact of integrating expression data with topological properties. We create the test set with new interactions from the 2018 version of BioGRID as positive interactions and the equal number of negative interactions randomly sampled. We make predictions for these interactions using learned embeddings and create a list of (Gene v_i, Gene v_j, probability), ranked by the predicted probability. We consider predicted gene pairs with the probabilities of 0.5 or higher but are missing from BioGRID for further investigation as we discuss later in this section.

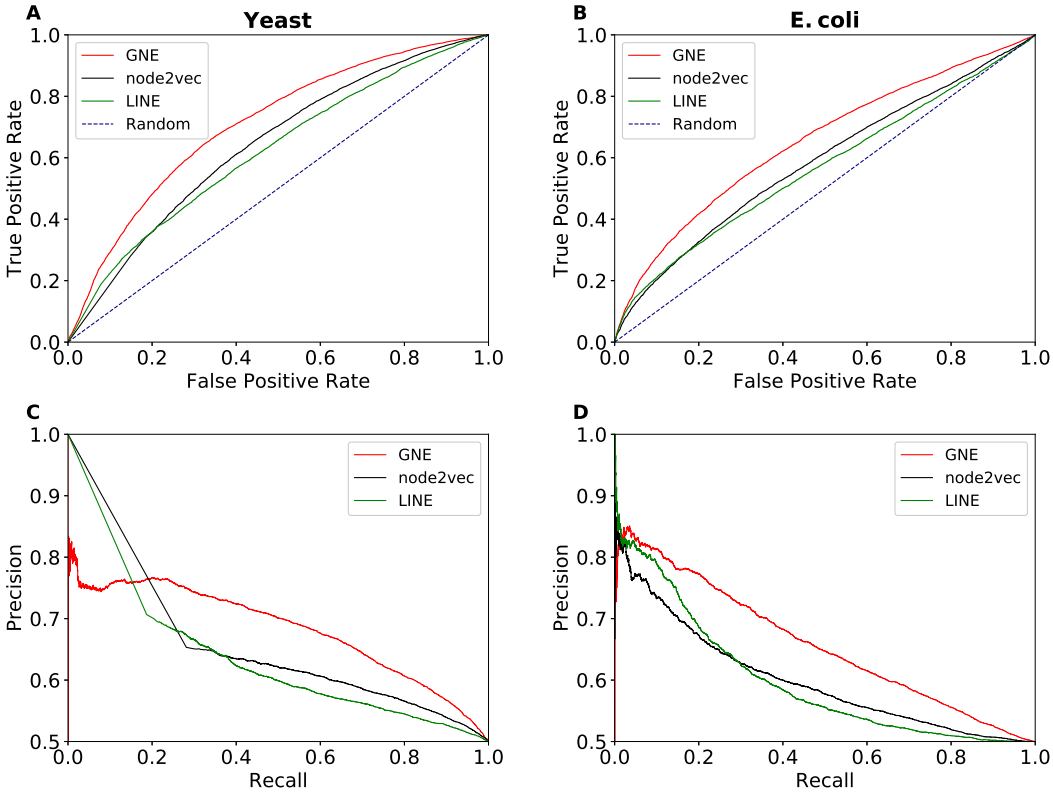


Figure 4.6: Temporal holdout validation in predicting new interactions. Performance is measured by the area under the ROC curve and the area under the precision-recall curve. Shown are the performance of each method based on the AUROC (**A, B**) and AUPR (**C, D**) for yeast and E. coli. The limit of the y-axis is adjusted to [0.5, 1.0] for the precision-recall curve to making the difference in performance more visible. GNE outperforms LINE and node2vec.

The temporal holdout performance of our model in comparison to other methods is shown in Figure 4.6. We observe that GNE outperforms both node2vec and LINE in temporal holdout validation across both yeast and E. coli datasets, indicating GNE can accurately predict new genetic

interactions. Table 4.5 shows that GNE achieves substantial improvement of 7.0 (AUROC), 7.4 (AUPR) on yeast and 6.6 (AUROC), 5.9 (AUPR) on E. coli datasets.

Methods	Yeast		E. coli	
	AUROC	AUPR	AUROC	AUPR
LINE	0.620	0.611	0.569	0.598
node2vec	0.640	0.609	0.587	0.599
GNE (our model)	**0.710**	**0.683**	**0.653**	**0.658**

Table 4.5: AUROC and AUPR comparision for temporal holdout validation. Note that method that achieves the best performance is bold faced.

Table 4.6 shows the top 5 interactions with the significant increase in predicted probability for both yeast and E. coli after expression data is integrated. We also provide literature evidence with experimental evidence code obtained from the BioGRID database [66] supporting these predictions. BioGRID compiles interaction data from numerous publications through comprehensive curation efforts. Taking new interactions added to BioGRID (version 3.4.158) into consideration, we evaluate the probability of these interactions predicted by GNE trained with and without expression data. Specifically, integration of expression data increases the probability of 8,331 (out of 11,185) interactions for E. coli (improving AUROC from 0.606 to 0.662) and 6,010 (out of 12,835) interactions for yeast (improving AUROC from 0.685 to 0.707). The integration of topology and expression data significantly increases the probabilities of true interactions between genes.

To further evaluate GNE's predictions, we consider the new version of BioGRID (version 3.4.162) and evaluate 2,609 yeast gene pairs (Additional file 1 Table S1) and 871 E. coli gene pairs (Additional file 1 Table S2) predicted by GNE with the probabilities of 0.5 or higher. We find that 128 (5%) yeast gene pairs and 78 (9%) E. coli gene pairs are true interactions that have been added to the latest release of BioGRID. We then evaluate the predictive ability of GNE by calculating the percentage of true interactions for different probability bins (Figure 4.7). 16% of predicted yeast gene pairs and 17.5% of predicted E. coli gene pairs with the probability higher than 0.9 are true interactions. This suggests that gene pairs with high probability predicted by GNE are more likely to be true interactions.

To support our finding that GNE predicted gene pairs have high value, we manually check gene pairs that have high predicted probability but are missing from the latest BioGRID release. We find that these gene pairs interact with the same set of other genes. For example, GNE predicts the interaction between YDR311W and YGL122C with a probability of 0.968. Mining the BioGRID

Organism	Probability		Gene i	Gene j	Experimental Evidence Code
	Topology	Topology + Expression			
Yeast	0.287	0.677	TFC8	DHH1	Affinity Capture-RNA [88]
	0.394	0.730	SYH1	DHH1	Affinity Capture-RNA [88]
	0.413	0.746	CPR7	DHH1	Affinity Capture-RNA [88]
	0.253	0.551	MRP10	DHH1	Affinity Capture-RNA [88]
	0.542	0.835	RPS13	ULP2	Affinity Capture-MS [89]
E. coli	0.014	0.944	ATPB	RFBC	Affinity Capture-MS [90]
	0.012	0.941	NARQ	CYDB	Affinity Capture-MS [90]
	0.013	0.937	PCNB	PAND	Affinity Capture-MS [90]
	0.015	0.939	FLIF	CHEY	Affinity Capture-MS [90]
	0.017	0.938	YCHM	PROB	Affinity Capture-MS [90]

Table 4.6: New gene interactions that are assigned high probability by GNE and the evidences supporting the existence of these predicted interactions.

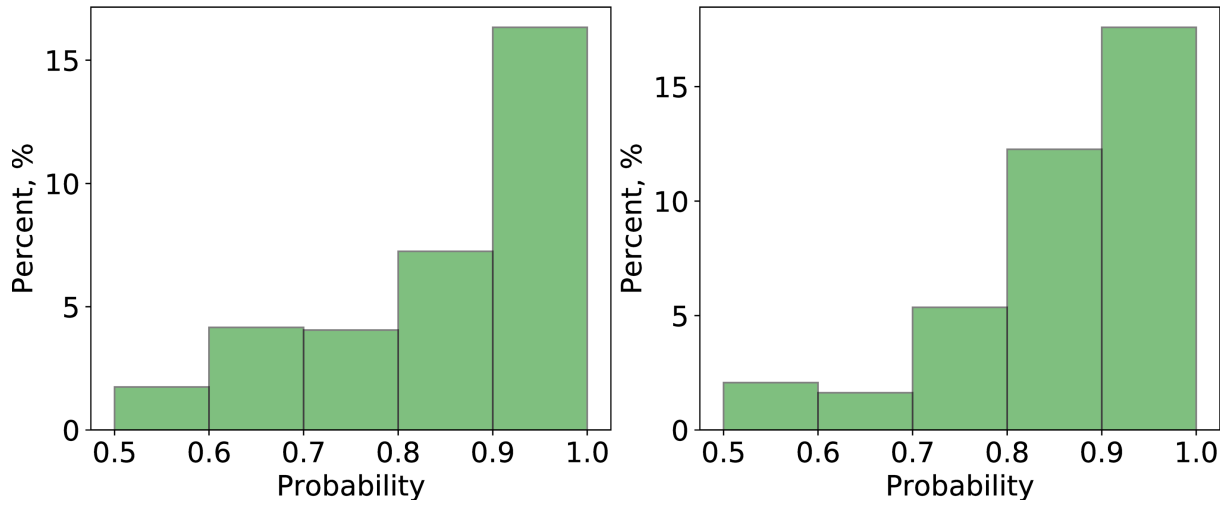


Figure 4.7: The percentage of true interactions from GNE's predictions with different probability bins. (a) yeast (b) E. coli. We divide the gene pairs based on their predicted probabilities to different probability ranges (as shown in the x-axis) and identify the number of predicted true interactions in each range. Each bar indicates the percentage of true interactions out of predicted gene pairs in that probability range.

database, we find that these genes interact with the same set of 374 genes. Similarly, E. coli genes

DAMX and FLIL with the predicted probability of 0.998 share 320 interacting genes. In this way, we identify all interacting genes shared by each of the predicted gene pairs in yeast and E. coli (Additional file 1 Table S1 and S2). Figure 4.8 shows the average number of interacting genes shared by a gene pair. In general, gene pairs with a high GNE probability tend to have a large number of interacting genes. For example, gene pairs with the probability greater than 0.9 have, on average, 82 common interacting genes for yeast and 58 for E. coli. Two sample t-test analysis has shown that there is a significant difference in the number of shared interacting genes for different probability bins (Table 4.7).

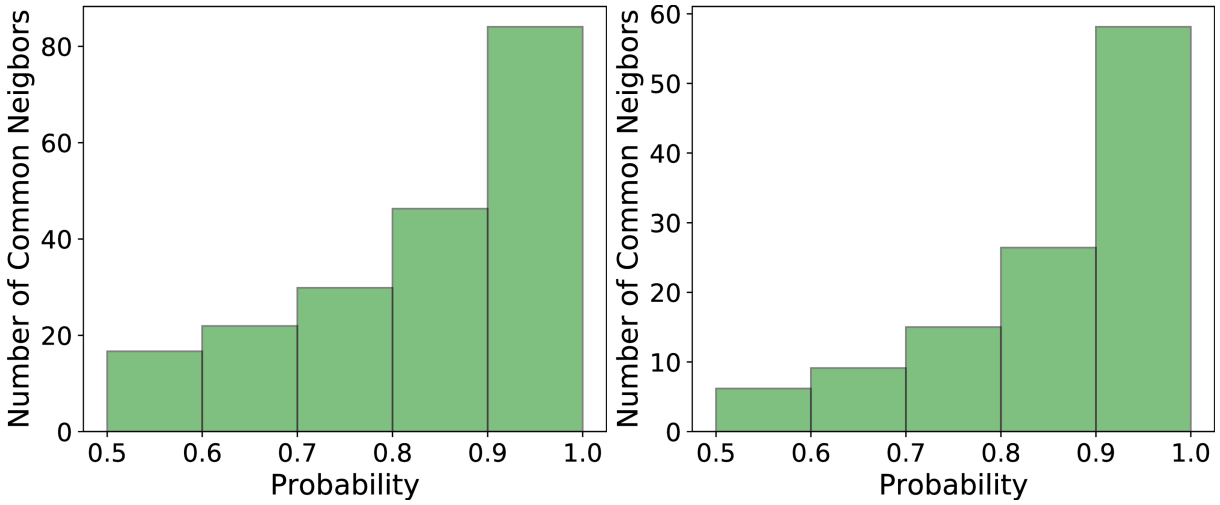


Figure 4.8: The average number of common interacting genes between the gene pairs predicted by GNE. (a) yeast (b) E. coli. We divide gene pairs into different probability groups based on predicted probabilities by GNE and compute the number of common interacting genes shared by these gene pairs. We categorize these gene pairs into different probability ranges (as shown in the x-axis). Each bar represents the average number of common interacting genes shared by gene pairs in each probability range.

Moreover, we search the literature to see if we can find supporting evidence for predicted interactions. We find literature evidence for an interaction between YCL032W (STE50) and YDL035C (GPR1), which has a probability of 0.98 predicted by GNE. STE50 is an adaptor that links G-protein complex in cell signaling, and GPR1 is a G-protein coupled receptor. Both STE50 and GPR1 share a common function of cell signaling via G-protein. Besides, STE50p interacts with STE11p in the two-hybrid system, which is a cell-based system examining protein-protein interactions [91]. Also, BioGRID has evidence of 30 physical and 4 genetic associations between STE50 and STE11. Thus, STE50 is highly likely to interact with STE11, which in turn interacts with

Probability bin for Sample A	Probability bin for Sample B	p-value for yeast	p-value for E. coli
0.5 - 0.6	0.6 - 0.7	$9.2e-14$	$6.9e-02$
0.5 - 0.6	0.7 - 0.8	$3.02e-51$	$1.23e-05$
0.5 - 0.6	0.8 - 0.9	$6.1e-117$	$7.4e-14$
0.5 - 0.6	0.9 - 1.0	$2.1e-177$	$3.7e-39$
0.6 - 0.7	0.7 - 0.8	$8.2e-17$	$1.1e-02$
0.6 - 0.7	0.8 - 0.9	$3.5e-69$	$9.2e-09$
0.6 - 0.7	0.9 - 1.0	$2.1e-128$	$1.9e-30$
0.7 - 0.8	0.8 - 0.9	$4.7e-28$	$4.8e-04$
0.7 - 0.8	0.9 - 1.0	$6.2e-87$	$7.4e-23$
0.8 - 0.9	0.9 - 1.0	$4.3e-35$	$5.1e-13$

Table 4.7: Results of two-sample t-test.

GPR1.

This analysis demonstrates the potential of our method in the discovery of gene interactions. Also, GNE can help the curator to identify interactions with strong potential that need to be looked at with experimental validation or within the literature.

4.4 Conclusion

We developed a novel deep learning framework, namely GNE to perform gene network embedding. Specifically, we design deep neural network architecture to model the complex statistical relationships between gene interaction network and expression data. GNE is flexible to the addition of different types and several attributes. The features learned by GNE allow us to use out-of-the-box machine learning classifiers like Logistic Regression to predict gene interactions accurately.

GNE relies on a deep learning technique that can learn the underlying patterns of gene interactions by integrating heterogeneous data and extracts features that are more informative for interaction prediction. Experimental results show that GNE achieves better performance in gene interaction prediction over other baseline approaches in both yeast and E. coli organisms. Also, GNE can help the curator to identify the interactions that need to be looked at.

Chapter 5

Representation Learning of biological networks using local network properties and sequential node features

In Chapter 4, we discussed the novel neural network for network representation learning that integrates continuous features with network structure. The proposed model takes node ID vector and node attributes as input to learn informative representation of nodes. The major limitation of the approach discussed in previous chapter is that these methods are not applicable to unseen nodes. In other words, this method can't make novel predictions for new nodes in the network since it requires node ID to be available during training and thus requires retraining whole model to make prediction for novel cases.

To address the problem, we design a novel neural network to learn representation of biological entities with only the node attributes as input and employing pairwise ranking loss function for optimization based on latent representation of entities. In this chapter, we focus on protein sequences as they are the most abundant and well-studied data available for proteins and thus consider to integrate protein sequences with interaction network. Furthermore, we propose to encode sequences as Gaussian distributions instead of vector representations to capture uncertainty associated with the representation. The proposed sparse regularization outputs the sparse mask that aligns with the biological motifs from pfam database, indicating the ability of our model to provide biological

insights to interpret the predictions.

5.1 Introduction

Proteins are the functional units within an organism that form molecular machines organized by their protein-protein interactions (PPIs) to carry out many biological and molecular processes. The primary structure of a protein, the protein sequence, determines the protein's unique three-dimensional shape, giving rise to an assumption that knowledge of the protein sequence alone might be sufficient to model the interaction between two proteins [92,93]. There is a longstanding interest in predicting PPIs from protein sequences which are by far the most abundant data available for proteins [94]. Traditional methods proposed to model PPIs involve extracting features based on domain expertise and training machine learning model on these features [94, 95, 96]. The performance of these methods relies heavily on the capability to select appropriate features, while the extracted features lack enough information about the interactions.

In this work, we propose a novel method to address these challenges. Specifically, we aim to learn a bidirectional GRU (BiGRU) model that maps variable-length sequences to a sequence of vector representations - one per amino acid position that encodes the sequential and contextual properties of amino acids. Since proteins interact with other proteins that perform different functions within a cell and even have different sequence patterns, we further encode the representation to the Gaussian distribution instead of a single point to capture uncertainty about its representation. We then define the cost function that incorporates the contrastive criteria between the latent Gaussian distributions such that the similarity between these distributions effectively captures complex interactions between proteins. It allows our model to minimize the statistical distance between interacting proteins while maximizing the distance for non-interacting proteins.

Alongside making accurate PPI predictions, it is crucial to have interpretable models that help domain experts understand how individual amino acids in the sequence contribute to the model's decisions. However, none of the state-of-the-art methods can provide such interpretability, limiting their practicality from biological perspectives. Since only a few amino acids in the interface region of the sequences are involved in interactions with other proteins [97], we, therefore, design a sparse and structured gate mechanism to guide the model to selectively focus on few amino acids in the sequence. The sparse gating mechanism outputs sparse weights - one per amino acid position - that explains how much contribution each amino acid makes and thus enhances interpretability.

Experimental results show that our method outperforms state-of-the-art methods on two challenging datasets: yeast and human PPIs from the BioGRID interaction database. Finally, we demonstrate that the sparse gate values learned by our model correspond to the biologically interpretable protein motifs. A literature-based case study illustrates that our model effectively learns to identify the important residues from the sequence.

5.2 Related works

Traditional methods focus on extracting features from protein sequences such as autocovariance (AC) [95], conjoint triads (CT) [93] and composition-transition-distribution (CTD) [94] descriptors and training a binary classifier on these features to predict PPIs [95, 96]. Since these extracted features only summarize the specific aspects of protein sequences such as physicochemical properties, frequencies of local patterns, and the positional distribution of amino acids, they lack enough information about the interactions.

Recently, deep learning architectures have been developed to address PPI prediction by automatically extracting useful features from protein sequences [98,99]. These methods adopt deep-Siamese like neural networks to model the mutual influence between protein sequences. They use encoder based on a convolutional neural network (CNN) to capture local features and recurrent neural network (RNN) to capture sequential and contextualized features from protein sequences. The encoder encodes a pair of sequences to lower-dimensional sequence vectors and a binary classifier predicts the probability of interaction based on these sequence vectors.

Specifically, DPPI [98] uses a deep-CNN based Siamese architecture that focuses on capturing local patterns from protein evolutionary profiles [100]. However, it requires extensive effort in data-preprocessing, specifically in constructing evolutionary profiles from protein sequences using Position-Specific Iterative BLAST (PSI-BLAST) [101]. To construct an evolutionary profile for a protein sequence, PSI-BLAST searches against NCBI non-redundant protein database with nearly 184 million sequences, which is time-consuming and makes it unscalable to a large number of protein sequences. However, PIPR [99] incorporates a deep residual recurrent convolutional neural network (RCNN) for PPI prediction using only the sequences of the protein pair. PIPR uses residual RCNN encoder that combines CNN to capture local features and RNN to capture sequential and contextualized features from protein sequences. The sequence representation for each protein is obtained by feeding its sequence through the deep sequence encoder with multiple layers of RCNN units. The non-intuitive mapping from protein sequences to their sequence representation makes

the representation difficult to interpret.

5.3 Method

A protein sequence $\mathbf{s}$ is a list of amino acids $[a_1, a_2, \ldots, a_L]$ where a_l is the amino acid at position l and L is the length of the sequence. We next formally define the problem of learning representation from protein sequences.

Definition 5.1. ***(Protein sequence Gaussian embedding)****: Using Definition 2.2, a protein-protein interaction network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with protein sequences as node attributes $\mathbf{X}$, we aim to map a protein sequence $\mathbf{s}$ to a lower-dimensional Gaussian distribution with formal format as follows: $f : \mathbf{s} \rightarrow (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} \in \mathbb{R}^d$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ with $d \ll L$ denoting the dimension of the Gaussian distribution.*

Next, we employ the Wasserstein distance as a measure of how different are the encoded Gaussian distributions of protein sequences. In this work, the goal of PPI prediction is to learn a model that predicts a smaller Wasserstein distance between the Gaussian distributions of interacting proteins and a larger difference for non-interacting proteins.

We introduce our deep learning framework for PPI prediction from sequences. The overall architecture of the proposed framework is illustrated in Figure 5.1. In step 1, the sequence encoder incorporates a bidirectional gated recurrent unit (BiGRU) to encode the amino acid sequence to sequence of vector representations. In step 2, the importance gate models dependencies between all the positions in the sequence, which could allow it to directly model residue-residue dependencies. This enables the model to compute the importance of amino acid in each position based on the sequence of vector representations. In step 3, the representation of sequence is encoded into Gaussian representation with mean $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. In step 4, a pairwise ranking loss is employed to minimize the statistical distance between interacting proteins while maximizing the distance for non-interacting proteins.

5.3.1 GRU-based sequence encoder

In step 1 as shown in Figure 5.1, the encoder takes a sequence of amino acids and encodes it to a sequence of vector representations - one per amino acid position. In particular, a one-hot encoded

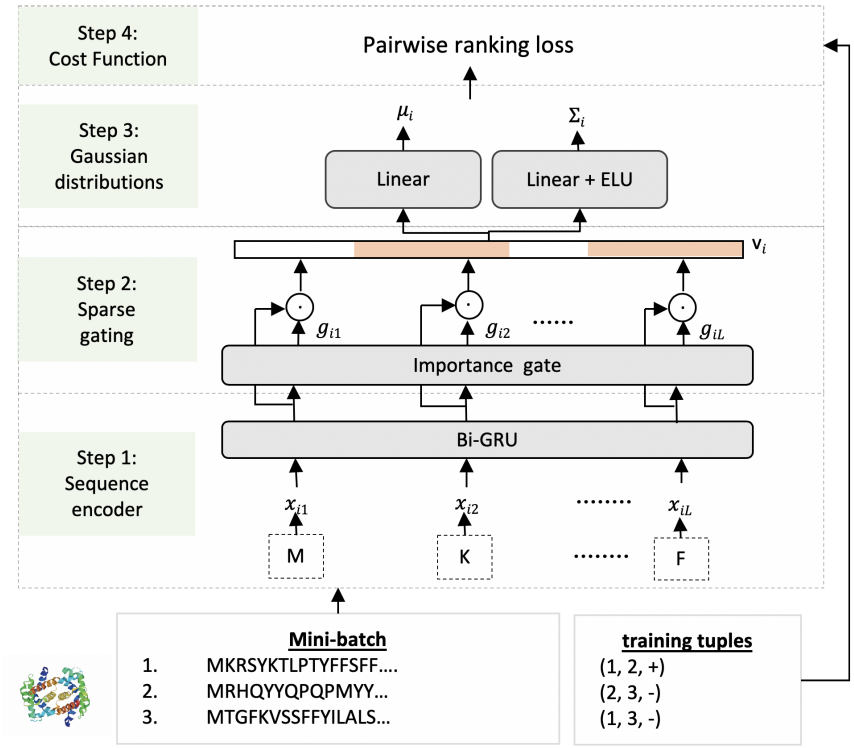


Figure 5.1: Diagram of the model. A mini-batch of protein sequence is encoded to Gaussian distribution $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using BiGRU encoder with sparse regularization. Model is trained by optimizing the pairwise Wasserstein distance between the training tuples (i.e. positive and negative interactions).

representation of amino acid $\mathbf{a}_l$ is embedded to a vector representation $\mathbf{x}_l$ through an embedding matrix:

$$\mathbf{x}_l = \mathbf{W}_e \mathbf{a}_l \tag{5.1}$$

where $\mathbf{W}_e \in \mathbb{R}^{N \times d}$ is the weight of the embedding layer.

To learn the sequential and contextualized representation of the amino acids in the sequence $\mathbf{s}$, we adopt the bidirectional Gated Recurrent Unit (BiGRU) that summarizes the sequence information from both directions. The sequence of vector representation $\mathbf{x}_l$ is then fed into the bidirectional gated recurrent unit. It contains two encoding processes: a forward encoding $\overrightarrow{\text{GRU}}$ which processes the sequence $\mathbf{s}$ from position 1 to L , and a backward encoding $\overleftarrow{\text{GRU}}$ which processes the sequence $\mathbf{s}$ from position L to 1.

$$\mathbf{h}_l = \text{BiGRU}(x_l) = [\overrightarrow{\text{GRU}}(\mathbf{x}_l), \overleftarrow{\text{GRU}}(\mathbf{x}_l)] \tag{5.2}$$

where $\mathbf{h}_l$ represents the GRU hidden state for amino acid $\mathbf{a}_l$. The representation of amino acid $\mathbf{h}_l$ is the concatenation of forward hidden state $\overrightarrow{\mathbf{h}}_l$ and backward hidden state $\overleftarrow{\mathbf{h}}_l$ which summarizes the information of whole sequence centered around $\mathbf{a}_l$.

5.3.2 Sparse gating mechanism

Proteins bind to each other at specific binding domains on each protein. These domains can be just a few peptides long or span hundreds of amino acids. For this purpose, we introduce additional gates $\mathbf{g} = \{g_1, g_2, \dots g_L\}$ indicating the activation of each amino acid. The amino acid a_l is active if $g_l > 0$ and is inactive when g_l is 0. The gate values g_l for the amino acid a_l, i.e., $g_l \in [0, 1]$ with 1 representing high importance. Let $\mathbb{S} = \{l | g_l > 0\}$ be the set of amino acid that are active indicated by their respective gate values. We obtain the representation for protein sequences by scaling the hidden state $\mathbf{h}_l$ with their respective gates:

$$\mathbf{v}_l = g_l \odot \mathbf{h}_l \tag{5.3}$$

$$\mathbf{v} = \text{Concat}_{(l \in \mathbb{S})}(\mathbf{v}_l) \tag{5.4}$$

where $\text{Concat}_{(l \in S)}$ represents the concatenation of sequence vectors for positions with $g_l > 0$. $\odot$ denotes the elementwise product between gate values g_l and GRU hidden state $\mathbf{h}_l$. The sparse gate values g_l leads to the sparse representation $\mathbf{v}$ of the sequence. In contrast, for the positions with $g_l = 0$, the hidden states $\mathbf{h}_l$ of these positions are not included in the representation $\mathbf{v}$. The sparse gates act as the controllers to selectively activate the part of the network to account only for the subset of amino acids of the sequence.

We introduce an auxillary network that takes the GRU hidden states $\mathbf{h}_{(.)}$ and generates the gate values for each position to determine whether the amino acid at that position is important for PPI prediction. The auxillary network models the long-range pairwise dependencies between amino acids in the sequence. With the auxillary network, our model explicitly considers dependencies between all position in the sequence, which could allow it to directly model residue-residue dependencies. In step 2 of Figure 5.1, GRU hidden states $\mathbf{h}_l$ is transformed to score p_l as:

$$p_l = \mathbf{W}_2(\tanh(\mathbf{W}_1 h_l + \mathbf{b}_1)) + \mathbf{b}_2 \tag{5.5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d}, \mathbf{W}_2 \in \mathbb{R}^{d\times 1}$, $\mathbf{b}_1 \in \mathbb{R}^d$ and $\mathbf{b}_2 \in \mathbb{R}^1$ are the weight matrices and biases for the linear layers. Let $\mathbf{p} = [p_1, p_2, \ldots, p_L]$ is a vector of scores for amino acids in the sequence $\mathbf{s}$. Next, we convert the vector of scores $\mathbf{p}$ to a probability distribution $\mathbf{g}$ so that $\sum_{l=1}^{L} g_l = 1$ and $0 \leq g_l \leq 1$. This allows us to quantify the relative contribution of each amino acid for the sequence representation. The softmax function is a simple choice to map the vector $\mathbf{p}$ to a probability distribution defined as:

$$\text{softmax}_i(\mathbf{p}) = \frac{\exp(p_i)}{\sum_j \exp(p_j)} \tag{5.6}$$

Since the resulting softmax distribution has full support, i.e., $\text{softmax}(p_l) > 0$ for every a_l, it forces all the amino acids in the sequence to receive some probability mass. Sotmax distribution assigns weights to each amino acid and even an unimportant amino acids have small weights, the weights on important amino acids become much smaller for long sequences, leading to degraded performance. However, not all of the amino acids in the sequence contribute towards the certain functions or interactions.

We introduce a sparsity regularization on $\mathbf{p}$ to only select the important sequence patterns, the set of amino acids, which may corresponds to the interface and may be important for interaction prediction. We employ sparsemax [102] as gate mechanism:

$$\text{sparsemax}(\mathbf{p}) := \underset{\mathbf{g}\in\Delta^K}{\text{argmin}} \|\mathbf{g} - \mathbf{p}\|^2, \tag{5.7}$$

where $\Delta^K := \{\in \mathbb{R}^K | \mathbf{g} \geq 0, \mathbf{1}^T\mathbf{g} = 1\}$ and $\mathbf{g}$ represents the Euclidean projection of $\mathbf{p}$ onto the $(K-1)$-dimensional probability simplex. The projection is likely to hit the boundary of the simplex, leading to sparse outputs, which allows the encoder to select only an informative subset of amino acids in the sequence. Although sparsemax leads to sparse representation, the sparse weights may only capture few important amino acids but may not identify a relatively long stretches of amino acids.

Furthermore, to enable our model to selectively focus on relatively longer stretches of amino acids, we present fusedmax regularization [103] that not only results in the sparse representations but also encourages the encoder to assign equal weights for contiguous sets of amino acids of the sequence while predicting interactions:

$$\text{fusedmax}(\mathbf{p}) := \underset{\mathbf{g}\in\Delta^K}{\text{argmin}} \frac{1}{2}\|\mathbf{g} - \frac{\mathbf{p}}{\gamma}\|^2 + \lambda \sum_{j=1}^{L-1} |\mathbf{g}_{j+1} - \mathbf{g}_j| \tag{5.8}$$

where γ controls the regularization strength and λ is the tuning parameter that balances the attention to produce sparse outputs and to assign equal weights to the amino acids within each segment. From Eq. 5.8, the first part projects the scores $\mathbf{p}$ to the probability simplex and the second part encourages paying equal attention to adjacent amino acids in the sequence. This allows the model to identify long stretches of amino acids that may bind with the residues from the interacting proteins

5.3.3 Gaussian representation of sequences

Within an organism, a given protein may be involved in a complex interplay with various proteins that perform different functions within a cell and even have different sequence patterns. Such differences should be reflected in the uncertainty of its representation $\mathbf{v}$. To model the uncertainty about the representation [104, 105], sequence representation $\mathbf{v}$ is then encoded to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in the final layer of the architecture as shown in step 3 in Figure 5.1. To ensure that covariance matrices $\boldsymbol{\Sigma}$ is positive definite, we use Exponential Linear Unit [84] in the final layer.

$$\boldsymbol{\mu} = \mathbf{W}_\mu \mathbf{v} + \mathbf{b}_\mu \tag{5.9}$$

$$\boldsymbol{\Sigma} = \text{ELU}(\mathbf{W}_\Sigma \mathbf{v} + \mathbf{b}_\Sigma) + 1 \tag{5.10}$$

where $\mathbf{W}_\mu, \mathbf{W}_\Sigma, \mathbf{b}_\mu$ and $\mathbf{b}_\Sigma$ denote the weight matrices and biases of linear layers that project intermediate representation $\mathbf{v}$ to mean μ and variance Σ of Gaussian representation of the sequence $\mathbf{s}$. We denote a protein sequence $\mathbf{s}$ with a d-dimensional Gaussian distribution $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ represents the center point of the sequence representation in the latent space and $\boldsymbol{\Sigma}$ represents the uncertainty associated with the representation. With an assumption that the different dimensions of the latent Gaussian distribution are independent of each other, the covariance matrix $\boldsymbol{\Sigma}$ is a diagonal matrix.

5.3.4 Loss function definition

We employ the Wasserstein distance to measure the similarity between the Gaussian distributions of the proteins to make PPI prediction. Wasserstein distance allows the model to capture the transitivity property of PPIs that measures the tendency of proteins to cluster together into functional modules and protein complexes [106].

The p^{th} Wasserstein distance between two probability measures μ and ν is defined as:

$$W_p(\mu,\nu)^p = \inf \mathbb{E}[d(\mathbf{X},\mathbf{Y})^p] \tag{5.11}$$

where $\mathbb{E}[\mathbf{Z}]$ denotes the expected value of a random variable $\mathbf{Z}$ and the infimum is taken over all joint distributions of random variables X and Y with marginals μ and ν respectively. Wasserstein distance is a well-defined measure that preserves both the symmetry and triangular inequality [107].

Wasserstein distance has a closed-form solution for two multivariate Gaussian distributions. This allows us to employ 2-Wasserstein distance (abbreviated as W_2) as similarity measure between the latent Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$ and $\mathcal{N}(\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)$:

$$dist = W_2(\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i),\mathcal{N}(\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)) \tag{5.12}$$

$$dist^2 = ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j - 2(\boldsymbol{\Sigma}_i^{\frac{1}{2}}\boldsymbol{\Sigma}_j\boldsymbol{\Sigma}_i^{\frac{1}{2}})^{\frac{1}{2}}) \tag{5.13}$$

Since we focus on diagonal covariance matrices, thus $\boldsymbol{\Sigma}_i\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}_j\boldsymbol{\Sigma}_i$:

$$dist^2 = ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2^2 + ||\boldsymbol{\Sigma}_i^{\frac{1}{2}} - \boldsymbol{\Sigma}_j^{\frac{1}{2}}||_F^2 \tag{5.14}$$

According to the above equation, the time complexity to compute the 2-Wasserstein distance (W_2) between two multivariate Gaussian distributions is linear with the embedding dimension d. Since the computation of W_2 no longer constitutes a computational challenge, we choose W_2 to measure the distance between latent representations.

We use a pairwise ranking formulation with respect to the Wasserstein distance W_2 to model PPIs:

$$W_2(\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i),\mathcal{N}(\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)) < W_2(\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i),\mathcal{N}(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)) \tag{5.15}$$

where $\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$ is the latent Gaussian distribution of sequence $\mathbf{s}_i$, and $(\mathbf{s}_i,\mathbf{s}_j)$ and $(\mathbf{s}_i,\mathbf{s}_k)$ represents

positive and negative interaction respectively. Specifically, the idea of ranking formulation is to penalize ranking errors based on the the Wasserstein distances between the pairs. The smaller the Wasserstein distance, the larger the possibility of interactions.

Finally, we employ square-exponential loss [108] to enable learning from the known pairwise interactions. Mathematically, the energy between the protein pairs can be defined as $E_{ij} = W_2(\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j))$. Then, the loss function to be optimized is:

$$\mathcal{L} = \sum_{i} \sum_{(i,j)\in\mathbb{Y}^+} \sum_{(i,k)\in\mathbb{Y}^-} ({E_{ij}}^2 + \exp(-E_{ik})) \tag{5.16}$$

where $\mathbb{Y}^+$ represents set of positive interactions and $\mathbb{Y}^-$ represents set of negative interactions. Positive interactions are the interactions with experimental evidence but negative interactions are randomly sampled from missing interactions. We can further increase the coverage of $\mathbb{Y}^+$ and $\mathbb{Y}^-$ by running random walk to generate node sequences and use a window to select positive interactions [15]. Furthermore, we can also follow similar node-based sampling strategy as [104]. For this experiment, we only consider the pairwise interactions from BioGRID database as positive interactions. In our setting, the objective function penalizes the pairwise errors by the energy of the pairs, such that the energy of positive interactions is lower than the energy of negative interactions. Equivalently, this will make the possibility of interactions between the interacting proteins larger than that of non-interacting proteins.

Finally, we can optimize the parameters Θ (i.e. weights and biases) of the model such that the loss $\mathcal{L}$ is minimized and the pairwise rankings are satisfied. Specifically, for each protein, the distance with interacting proteins should be smaller than with non-interacting proteins. We term this as the ranking approach since interacting proteins have smaller W_2 distance and are ranked higher than non-interacting proteins.

5.3.5 PPI prediction

The Wasserstein distances between the latent Gaussian distributions of protein sequences corresponds to the possibility of their interaction. However, predicting PPIs by only computing the Wasserstein distance fails to take into account the homodimers, the proteins with identical sequences [98]. The encoded Gaussian representations of these protein sequences will be the same and their Wasserstein distance will be 0 indicating they must interact.

To overcome this limitation, we define pairwise features for all protein pairs by the concatenation of

the absolute element-wise differences of means and variances and the element-wise multiplications of the means of their Gaussian representations, $[|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|; |\boldsymbol{\Sigma}_i - \boldsymbol{\Sigma}_j|; \boldsymbol{\mu}_i \odot \boldsymbol{\mu}_j]$. This featurization is effective in modeling the symmetric relationship between proteins. To predict binary interactions, we train a binary classifier on these pairwise features to learn a decision boundary δ that separates interacting proteins from non-interacting pairs.

5.3.6 Efficient training

Siamese networks are suitable to train with contrastive loss mentioned in Eq. 5.16. However, it is inefficient to train Siamese networks when the amount of PPIs increases. In particular, the possible number of interactions for N proteins is $(N^2 + N)/2$ (including self-interactions), which is computationally intensive to train with Siamese architecture. A mini-batch of m interactions in Siamese training may have multiple occurrences of the same proteins, leading the protein sequence to be feed-forwarded for each interaction. It is sufficient to feed-forward each protein sequence once to compute the loss in the batch. To address this problem, we encode the minibatch of n protein sequences and retrieve positive and negative interactions that involve them to compute the loss in the minibatch. With this setting, we are only required to feed-forward N protein sequences compared to $(N^2 + N)/2$ pairs in the Siamese setting, which makes our method computationally efficient and scalable to a large number of interactions.

5.4 Experiments

We evaluate our method on two real-world datasets: yeast and human proteins to predict their interactions. We use the area under the ROC curve (AUROC) and the average precision (AP) scores as the evaluation metrics. With these evaluation metrics, we expect the positive protein pair to have higher interaction probability compared to negative protein pair.

5.4.1 Experimental setup

Datasets

The datasets for protein sequences of yeast and human proteins are from the EMBL-EBI Reference Proteome [109]. The information about the subcellular localization of proteins is extracted from

UniProt database [110]. The evolutional protein profiles for yeast and human protein sequences are collected from Rost Lab [111]. We evaluate our proposed model with two types of protein features: (a) amino acid sequences and (b) evolutionary protein profiles constructed from these sequences.

To evaluate the performance of deep learning models, the interaction datasets are downloaded from the up-to-date BioGRID interaction database (Release 3.5.169) [112]. The BioGRID database provides a large number of PPIs allowing us to evaluate the scalability of different approaches as well. Only the interactions that correspond to the physical binding between the protein pairs (say $\mathbb{Y}^+$) are considered since these interactions are supported by experimental evidence. The negative samples $\mathbb{Y}^-$ are generated by randomly sampling from all protein sequence pairs ($(\mathbf{s}_i, \mathbf{s}_j) \notin \mathbb{Y}^+$), that are not yet confirmed by experimental evidence. Furthermore, these negative interactions are filtered based on their subcellular localization, assuming that proteins in the different locations are unlikely to interact although some proteins do translocate [111].

Also, we perform the cluster analysis with the CD-HIT [113] to cluster protein sequences based on a certain similarity threshold that represents sequence identity. We remove the interactions such that no two proteins have a pairwise sequence identity greater than 10%. Table 5.1 shows the statistics of the interaction datasets.

Data	No. of proteins	No. of positive pairs	No. of negative pairs
Yeast	3,651	50,344	50,376
Human	7,028	73,624	73,628

Table 5.1: Statistics of interaction datasets

Hyperparameters and training details

For both datasets, we train a sequence encoder with the same configuration. The best hyperparameters of our model are selected based on validation performance. The maximum length of the input sequence to the encoder is 1024 for efficient training. In the datasets, 91.2% of yeast sequences and 86% of human sequences are shorter than 1024 residues.

The encoder consists of a BiGRU layer with 16 hidden units each, to map a protein sequence to a sequence of 32-dimensional representation, one per amino acid. Then, this representation is encoded to a latent Gaussian distribution with dimension $\boldsymbol{d} = 256$ ($\boldsymbol{d} = 2 \times d = 128$ for the mean

Method		Data	Yeast		Human	
			AUROC	**AP**	**AUROC**	**AP**
DPPI [98]		Profiles	0.891 ±0.004	0.857 ±0.007	0.870 ±0.004	0.835 ±0.005
PIPR [99]		sequences	0.909 ±0.003	0.912 ±0.004	0.878 ±0.002	0.882 ±0.003
Our method (sparsemax)	Ranking	Profiles	0.882 ±0.003	0.888 ±0.002	0.884 ±0.003	0.893 ±0.004
		Sequences	0.901 ±0.002	0.904 ±0.002	0.881 ±0.002	0.889 ±0.001
	Random Forest	Profiles	0.908 ±0.002	0.913 ±0.003	**0.891 ±0.005***	**0.896 ±0.005***
		Sequences	**0.924 ±0.002***	**0.925 ±0.001***	0.887 ±0.002	0.894 ±0.001
Our method (fusedmax)	Ranking	Profiles	0.882 ±0.006	0.885 ±0.006	0.873 ±0.09	0.881 ±0.01
		Sequences	0.898 ±0.001	0.900 ±0.002	0.874 ±0.002	0.883 ±0.001
	Random Forest	Profiles	0.906 ±0.004	0.912 ±0.005	0.872 ±0.015	0.877 ±0.015
		Sequences	0.919 ±0.003	0.921 ±0.002	0.881 ±0.002	0.886 ±0.001

Table 5.2: Average AUROC and AP scores (with standard deviation) averaged over five independent runs for PPI prediction. * represents statistically significant differences with PIPR (P-value $<$ 0.005).

and 128 for the variance of the Gaussian).

All the weight matrices of the encoder layer are initialized using Xavier initialization [114]. The model is trained on a single NVIDIA GeForce RTX 2080 Ti GPU for all experiments using Adam optimizer [63] with learning rate 0.003 and other default parameters provided by PyTorch [115]. Our unique approach of encoding unique sequences allows us to train the model efficiently even with large batch sizes. We empirically find that our model converges in a small number of iterations ($\leq$ 50 for all shown experiments).

5.4.2 Results on PPI prediction

We compare our method against the state-of-the-art deep learning methods on the up-to-date BioGRID interaction datasets. We split the interactions into training, validation, and test sets (0.6:0.2:0.2). All the models are trained on the same training set and the best set of hyperparameters are selected based on their performances on the validation set. Finally, the models are evaluated on independent test sets. Table 5.2 reports the mean AUROC and AP and their standard errors on five independent runs. We perform a two-tailed Welch's t-test and Benjamini-Hochberg procedure to adjust p-value and find that the improvement over PIPR, the state-of-the-art method is statistically significant.

With the ranking approach, we expect our model to rank positive interactions higher than negative interactions, i.e. the probability of interactions between interacting protein pairs is greater than that of non-interacting protein pairs. Table 5.2 demonstrates that our model ranks positive interactions higher than negative interactions. The ranking based model with sparsemax regularization achieves comparable performance with PIPR. Furthermore, the random forest classifier trained to account for homodimeric interactions improve the model's performance on both datasets. The best parameters for random forest classifier are selected via grid search.

Furthermore, we evaluate our proposed method on evolutionary protein profiles. Protein profiles constructed from protein sequences capture the correlation between different proteins as well as between different parts of the sequences [116]. Our model trained with protein profiles outperforms DPPI, the state-of-the-art deep architecture that uses profiles as the input to their deep Siamese model. Table 5.2 shows that our model with sequences achieves comparable or better performance compared to profiles across both datasets. This demonstrates that our model is capable of extracting useful information about interactions from protein sequences and alleviates the expensive process of profile construction from sequences.

5.4.3 Ablation study of framework components

We next evaluate individual model components on the PPI prediction task with yeast dataset.

Gaussian representations outperform point representations

We first explore whether the Gaussian representation of sequences improves the performance of the model over point representation. For this experiment, we encode intermediate representation $\mathbf{v}$ of sequence $\mathbf{s}$ to point representation $\mathbf{z} = \mathbf{W_z v} + \mathbf{b_z}$ instead of Gaussian representation (as in Eq. 5.9 and 5.10) and define L2 norm as the similarity between the point representations of sequences $\mathbf{s}_i$ and $\mathbf{s}_j$ instead of Wasserstein distance (in Eq. 5.12):

$$dist^2 = ||\mathbf{z}_i - \mathbf{z}_j||_2^2 \tag{5.17}$$

where $\mathbf{z}$ represents point representation of sequences s. Table 5.3 shows that Gaussian representations are better predictors of PPIs compared to point representations.

Model configuration		AUROC	AP
No gating		0.880±0.001	0.875±0.003
Point + RF	Softmax	0.881±0.001	0.877±0.001
	Fusedmax	0.909±0.001	0.912±0.002
	Sparsemax	0.913±0.001	0.916±0.002
Gaussian + RF	Softmax	0.882±0.001	0.879±0.002
	Fusedmax	0.919±0.003	0.921±0.001
	Sparsemax	**0.924±0.002**	**0.925±0.001**

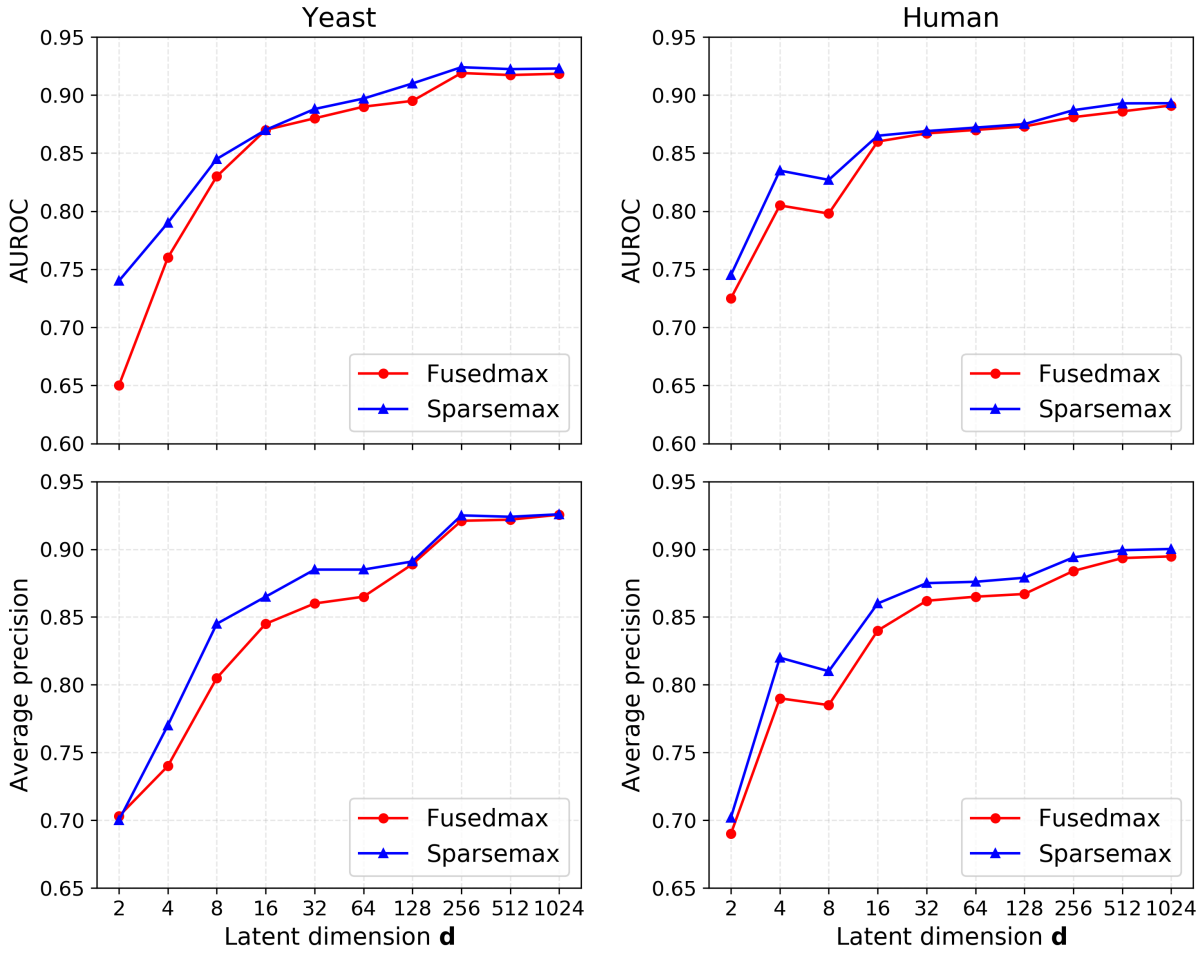
Table 5.3: Study of individual model components on Yeast dataset. The model trained without any gate mechanism (No gating), with different representations (point vs Gaussian) coupled with different regularization strategies and random forest (RF) classifier.

Sparse gating mechanism improves performance

We further demonstrate the importance of the proposed sparse gating mechanism by comparing the performance of the sequence encoder trained with and without various gating mechanism. We train models with different settings of the gate mechanism. Table 5.3 demonstrates that the sparse gating mechanism provides a significant improvement over no gate mechanism and softmax in PPI prediction. This indicates that sparse regularization helps the model to selectively activate the important stretches of amino acids that are important to model and predict PPIs.

Dimension of Gaussian distribution is important

Finally, we investigate how the dimensionality of the latent Gaussian distributions can affect the model's performance. Figure 5.2 shows the plot of the AUROC and the AUPR scores of our method across the two organisms. When the dimension $\boldsymbol{d}(= 2 \times d)$ of the Gaussian distribution increases from 2 to 256, the performance also increases. When $\boldsymbol{d} \geq 128$, two regularization strategies result in similar performance. Moreover, the performance also remains stable when $\boldsymbol{d} \geq 256$.



(a) (b)

Figure 5.2: Effect of dimension of latent Gaussian distributions on the model's performance with different sparse gating mechanism for (a) yeast (b) human.

5.4.4 Training time comparison

Here, we compare the training time of our method and PIPR, the state-of-the-art deep learning model [99]. We report the average training time per epoch in Figure 5.3. For the fair comparison, we train both models in the same machine on the same dataset and compare only the average training time per epoch. For this experiment, we randomly sample 8k, 16k, 24k, 32k, 40k, 48k, 56k, 64k, 72k, 80k, and 88k training interactions.

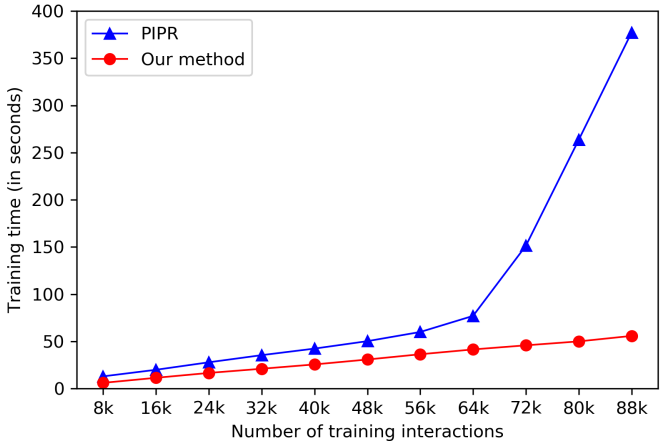


Figure 5.3: Average training time per epoch. Our method is much faster than PIPR. Encoding the unique set of protein sequences to their latent Gaussian distribution and optimizing the loss based on their interactions makes our model efficient.

Figure 5.3 shows that our method is efficient in comparison to PIPR. PIPR uses a pairwise training process that requires a higher number of matrix multiplications for each interaction. On the other hand, given the large batch of interactions, our model finds the unique set of protein sequences involved in these interactions and encodes them, which significantly reduces the number of matrix multiplication. Once we have the embeddings for these sequences, we can compute the loss based on their respective interactions. As discussed in Section 5.3.6, for instance, if there are 1000 interactions in a batch with 100 unique proteins, our model encodes only 100 protein sequences instead of 1000 protein pairs as in PIPR. Note that this approach allows our model to train on a large batch of interactions, and thus takes less time to train. This demonstrates that our method scales with the number of interactions and is significantly faster than other sophisticated models such as PIPR.

5.4.5 Qualitative evaluation

Since our proposed model selectively activates the part of a given sequence, it is important to evaluate whether the selected parts are important. To explore this, we perform the quantitative evaluation on how the amino acids selected by the sparse gating mechanism in our proposed method align with the motifs from the Pfam motif library [27] from GenomeNet [117]. The gate vector $\mathbf{g}$ for a sequence $\mathbf{s}$ helps us interpret how much contribution an amino acid on that position signaled by GRU hidden state makes. Since our model computes gate values between 0 and 1 for each amino acid, we consider amino acids with $g_l > 0$ to be active i.e. used by the model for the representation of proteins. Table 5.4 shows the average percentage of amino acids selected by the sparse gating mechanism and their alignment with motifs having biological significance. For instance, for yeast dataset, only 19.24% amino acids (on average) are selected with fusedmax and 59.05% of these selected amino acids aligns with the motif. This illustrates that the amino acids in the sequence selectively activated by our model to learn protein representation align with biologically interpretable motifs.

Dataset	Gating	Selected amino acids (%)	Alignment with motifs (%)
Yeast	Sparsemax	8.06	49.96
	Fusedmax	19.24	59.05
Human	Sparsemax	9.15	48.33
	Fusedmax	23.33	65.63

Table 5.4: Comparison of selected amino acids with the motifs from Pfam motif database

In addition, we visualize the amino acids selected by our model and the motifs from Pfam motif library [27] from GenomeNet [117] in Figure 5.4. For this experiment, we select three proteins: LSM8, SMD2, and RPC11 from the yeast dataset with motifs in different parts of the sequences. We train the model with fusedmax and obtain the gate values for each amino acid in these sequences. Red lines in each subfigure of Figure 5.4 are the regions identified to be important by our model.

In particular, LSM8 with the sequence of length 109 has two motifs: PF01423, LSM domain at the position from 4 to 65, and PF14807, adaptin AP4 complex epsilon appendage platform at the position from 9 to 65 shown in Figure 5.4a. The gate value learned by our model corresponds to the subset of amino acids from position 1 to 65 and aligns with motifs. Similarly, the selected parts of sequences for SMD2 and RPC11 aligns with their motifs even though the motif lies in different parts of sequences. The quantitative and qualitative evaluation of gate vectors shows that our model successfully identifies important amino acids in the sequence for PPI prediction.

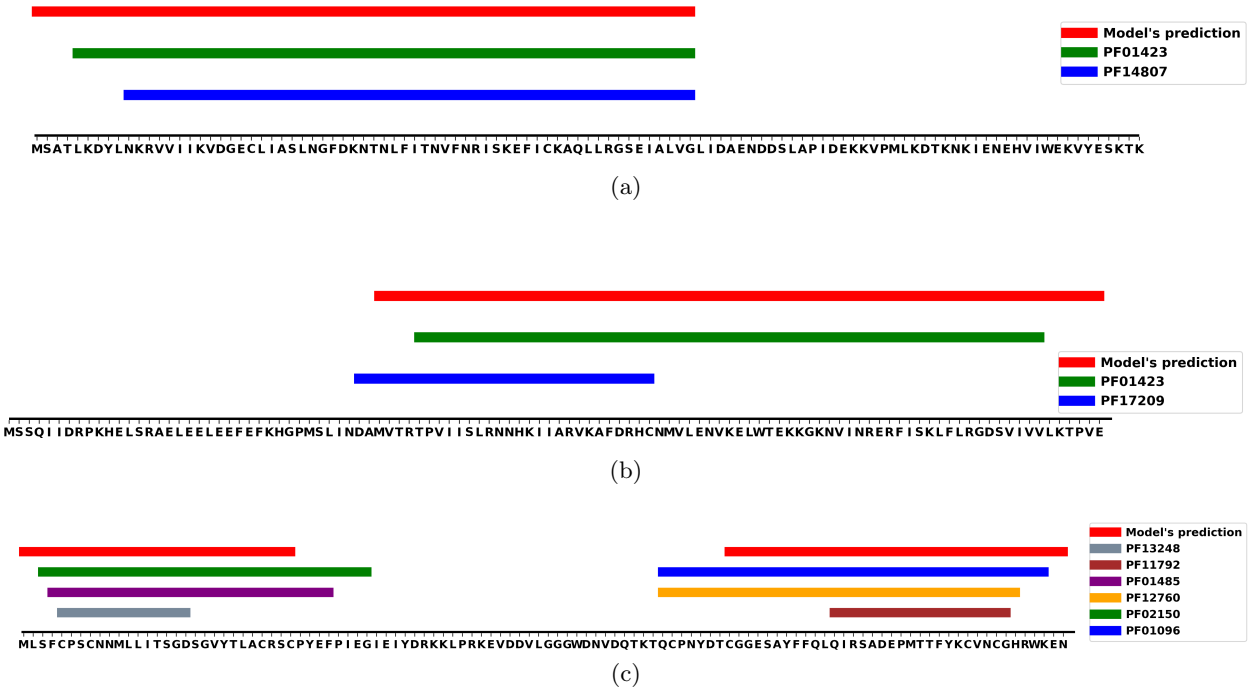
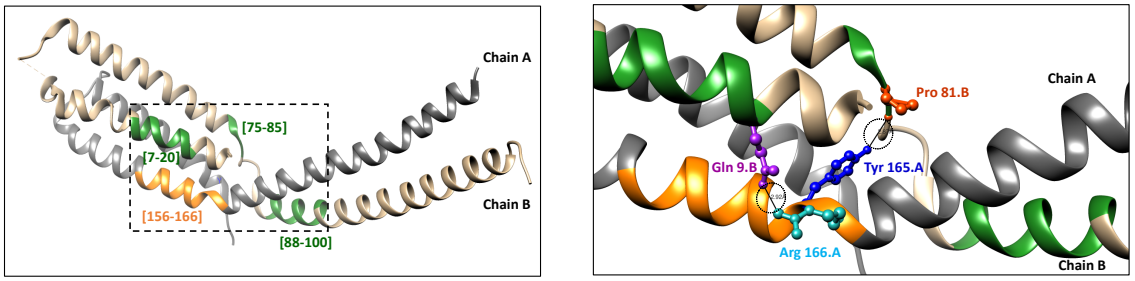


Figure 5.4: Visualization of motifs and the amino acids selected by our model for three proteins (a) LSM8 (b) SMD2 and (c) RPC11. The legend includes the motif identifier from Pfam database. The alphabet on the x-axis is the amino acid at the respective position. The line segments corresponds to the part of sequence that is model's prediction (red) or motifs (other color). The selected subset of amino acid sequences aligns with motifs from Pfam motif library.

Case study for interpretability

Here, we evaluate whether there exists hydrogen bonds between the interacting protein pairs in the region selected by our model. For this experiment, we select a protein complex 1YKE, the mediator MED7/MED21 subcomplex [118] that contains two interacting proteins: Q08278 and P47822. The interaction between these proteins is supported by experimental evidence. We exclude the interaction between Q08278 and P47822 from the training interactions and train our model with fusedmax regularization. We obtain the gate vectors that corresponds to the region selected by our model for both proteins. We visualize the 3D structure of the complex using Chimera [119] in Figure 5.5. Also, the selected residues are highlighted with green and yellow in the protein chains based on their respective gate vectors from the trained model.

Figure 5.5(a) shows the residues selected by our method in the chains of protein complex 1YKE.



(a) (b)

Figure 5.5: Important residues predicted by our method in mediator MED7/MED21 subcomplex (1YKE) interface. (a) Only chain A (Q08278) and chain B (P47822) are shown, in which the important regions predicted by our method are shown in yellow and green, respectively. (b) Shown is the zoom-in view of the area surrounded by dashed lines in the 1YKE structure. Two hydrogen bonds (shown in black lines marked by black circle) are formed between chain A and B.

Our method selects the amino acids from position 7 to 20, 75 to 85, and 88 to 100 for P47822 and position 156 to 166 for Q08278. The importance of these amino acids for PPIs are clearly illustrated in the structure. There is at least one hydrogen bond between the regions selected in these sequences (chain A and B). Specifically, Glutamine (Gln) at position 9 (shown in purple) and Proline (Pro) at position 81 (shown in red) of chain B (P47822) has a hydrogen bond with Arginine (Arg) at position 166 (shown in cyan) and Tyrosine (Tyr) at the position of 165 (shown in blue) of chain A (Q08278) respectively (Figure 5.5(b)). Thus, our model provides reasonable cues on the important interface residues for the PPI.

5.5 Conclusion

We present a novel deep neural network to model and predict PPIs from variable-length protein sequences. Our proposed framework adopts a recurrent neural network to capture contextualized and sequential information from the primary amino acid sequences. By incorporating a structured and sparse gating mechanism into the sequence encoder, our model successfully selects the important residues on the sequence to learn the sequence representation. Furthermore, our novel approach of encoding sequences to their representation makes the model efficient and scalable to a large number of interactions. Extensive experimental evaluations on various up-to-date datasets show its promising performance on binary PPI prediction task. Various case studies demonstrate the ability of our model to provide biological insights to interpret the predictions.

Chapter 6

Representation Learning of biological networks using higher-order network properties

In previous chapters, we developed network representation learning methods for interaction prediction by integrating network properties with additional information such as gene expression data, protein sequences. However, these methods only consider the local neighborhood information with first-order and second-order proximity. Since the effect on one molecular entity is propagated via pathways to form certain products or give rise to emergent characteristics of the biological system, the model that only captures local network structure cannot give a comprehensive view of the biological system.

In this chapter, we discuss that the model capable of capturing information from farther neighborhood is crucial to capture the global context of the entities in the biological network. To this aim, we propose a novel deep learning framework that aggregates information from various neighborhood to learn representation for each biological entity in the network. We evaluate the proposed model across datasets such as protein-protein interactions (PPIs) [120], drug-drug interactions (DDIs) [121], drug-target interactions (DTIs) [122] and gene-disease associations (GDIs) [123] with varying number of nodes and edges to demonstrate its effectiveness and scalability.

6.1 Introduction

Recently, the generalization of deep learning to the network-structured data [46] has shown great promise across various domains such as social networks [124], recommendation systems [125], chemistry [126], citation networks [43]. These approaches are under the umbrella of graph convolutional networks (GCNs). GCNs repeatedly aggregate feature representations of immediate neighbors to learn the informative representation of the nodes for link prediction. Although GCN based methods show great success in biomedical interaction prediction [24, 121], the issue with such methods is that they only consider information from immediate neighbors. SkipGNN [19] leverages skip graph to aggregate feature representations from direct and second-order neighbors and demonstrated improvements over GCN methods in biomedical interaction prediction. However, SkipGNN cannot be applied to aggregate information from higher-order neighbors and thus fails to capture information that resides farther away from a particular interaction [28].

To address the challenge, we propose an end-to-end deep graph representation learning framework named higher-order graph convolutional networks (HOGCN) for predicting interactions between pairs of biomedical entities. HOGCN learns a representation for every biomedical entity using an interaction network structure $\mathcal{G}$ and/or features $\mathbf{X}$. In particular, we define a higher-order graph convolution (HOGC) layer where the feature representations from k-order neighbors are considered to obtain the representation of biomedical entities. The layer can thus learn to mix feature representations of neighbors at various distances for interaction prediction. Furthermore, we define a bilinear decoder to reconstruct the edges in the input interaction network $\mathcal{G}$ by relying on feature representations produced by HOGC layers. The encoder-decoder approach makes HOGCN an end-to-end trainable model for interaction prediction.

We compare HOGCN's performance with that of state-of-the-art network similarity-based methods [127], network embedding methods [15, 40], and graph convolution-based methods [19, 43, 44] for biomedical link prediction. We experiment with various interaction datasets and show that our method makes accurate and calibrated predictions. HOGCN outperforms alternative methods based on network embedding by up to 30%. Furthermore, HOGCN outperforms graph convolution-based methods by up to 6%, alluding to the benefits of aggregating information from higher-order neighbors.

We perform a case study on the DDI network and observe that aggregating information from higher-order neighborhood allows HOGCN to learn meaningful representation for drugs. Moreover, literature-based case studies illustrate that the novel predictions are supported by evidence,

suggesting that HOGCN can identify interactions that are highly likely to be a true positive.

In summary, our study demonstrates the ability of HOGCN to identify potential interactions between biomedical entities and opens up the opportunities to use the biological and physicochemical properties of biomedical entities for a follow-up analysis of these interactions.

6.2 Related works

With the increasing coverage of the interactome, various network-based approaches have been proposed to exploit already available interactions to predict missing interactions [17, 127, 128, 129]. These methods can be broadly classified into (1) network similarity-based methods (2) network embedding methods (3) graph convolution-based methods. We next summarize these categories of methods for biomedical interaction prediction.

Given a network of known interactions, various similarity metrics are used to measure the similarity between the biomedical entities [128] with an assumption that higher similarity indicates interaction. Triadic closure principle (TCP) has been explored in biomedical interaction prediction with the hypothesis that biomedical entities with common interaction partners are likely to interact with each other [129]. TCP relies on a common neighbor algorithm to count the number of shared neighbors between the nodes and is quantified by $\mathbf{A}^2$ where $\mathbf{A}$ is the adjacency matrix. Recently, L3 heuristic [127] shows the common neighbor hypothesis fails for most protein pairs in PPI prediction and proposes to consider nodes that are similar to the neighbors of the nodes and can be quantified by $\mathbf{A}^3$. This indicates that higher-order neighbors are important for interaction prediction.

Next, network embedding methods embed the existing networks to low-dimensional space that preserves the structural proximity such that the nodes in the original network can be represented as low-dimensional vectors. Deepwalk [40] is a popular approach that generates the truncated random walks in the network and defines a neighborhood for each node as a set of nodes within a window of size k in each random walk. Similarly, node2vec performs a biased random walk by balancing the breadth-first and depth-first search in the network. The random walks generated by these methods can be considered as a combination of nodes from various order of neighborhoods such as 1-hop to k-hop neighborhood. In other words, DeepWalk and node2vec learn the embeddings for the nodes in the network from the combination of $\mathbf{A}^1, \mathbf{A}^2, \mathbf{A}^3, \ldots \mathbf{A}^k$ where $\mathbf{A}^i$ is the i^{th} power of the adjacency matrix. These embeddings can then be fed into a classifier to predict the interaction probability. These methods are only limited to the structure of the biomedical networks and cannot incorporate

additional information about the biomedical entities. Also, they cannot learn the feature difference between nodes at various distances.

Furthermore, graph convolution-based methods use a message-passing mechanism to receive and aggregate information from neighbors to generate representations for the nodes in the network. Graph convolutional networks (GCNs) [44] and variational graph convolutional autoencoder (VGAE) [43] aggregate feature representation from immediate neighbors to learn the representation of biomedical entities in an end-to-end manner using link prediction objective. These methods are only limited to the average pooling of the neighborhood features [28]. SkipGNN [19] therefore proposes to use skip similarity between the biomedical entities to aggregate information from second-order neighbors. However, these methods cannot aggregate feature representations from higher-order neighbors and also cannot learn feature differences between neighbors at various distances.

6.3 Preliminaries

A biomedical network is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ where $\mathcal{V}$ denotes the set of nodes representing biomedical entities (e.g. proteins, genes, drugs, diseases) and $|\mathcal{V}|$ denotes the number of nodes. $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$ denotes a set of interactions between biomedical entities. $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times F}$ is the features of biomedical entities where F is the dimension of features.

Let $\mathbf{A}$ denote the adjacency matrix of $\mathcal{G}$, where A_{ij} indicates an edge between nodes v_i and v_j. We assume the case of binary adjacency matrix $A_{ij} \in \{0, 1\}^{n \times n}$ where A_{ij} represents the existence of edge between the nodes v_i and v_j, indicating the presence of the experimental evidence for their interaction (i.e. $A_{ij} = 1$) or the absence of the experimental evidence for their interaction (i.e. $A_{ij} = 0$). Note that the same notation of adjacency matrix can be used to represent weighted graphs such that $A_{ij} = [0, 1]$. Table 6.1 shows the notations and their definitions used in the paper.

6.3.1 Message passing

Given a biomedical network $\mathcal{G}$, message passing algorithms learn the representation of biomedical entities in the network by aggregating information from immediate neighbors [126]. Additional information about biomedical entities can be used to initialize the feature matrix $\mathbf{X}$. These algorithms involve the message passing step in which each biomedical entity sends its current representation to,

Notation	Definition
$\mathcal{G} : \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$	Graph with nodes $\mathcal{V}$, edges $\mathcal{E}$ and features $\mathbb{P}$
$\mathcal{E}'$	Test edges
$\mathbf{A} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$	Adjacency matrix of graph $\mathcal{G}$
$\mathbf{D} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$	Degree matrix with $D_{ii} = \sum_i A_{ij}$
$\mathbf{I} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$	Identity matrix
$\hat{\mathbf{A}} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$	Symmetrically normalized adjacency matrix
$\mathbf{X} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times F}$	F-dimensional feature matrix
$A_{ij} \in \{0, 1\}$	Ground-truth interaction between nodes i and j
$p_{ij} \in [0, 1]$	Probability of interaction between nodes i and j
$\mathbf{Z} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d^*}$	Final node embeddings
$\mathbf{W}_{(l)}^{(i)}$	Weight matrix for i^{th} adjacency power for layer l
L	Number of HOGC layers
T	Number of training epochs
k	The order of neighborhood
$\mathbb{P}$	A set of integer adjacency powers $\mathbb{P} = \{0, 1, \ldots, k\}$
$\mathbf{O}_j^{(l)}$	Representation of neighbors at distance j in layer l

Table 6.1: Terms and notations for higher-order graph convolutional network

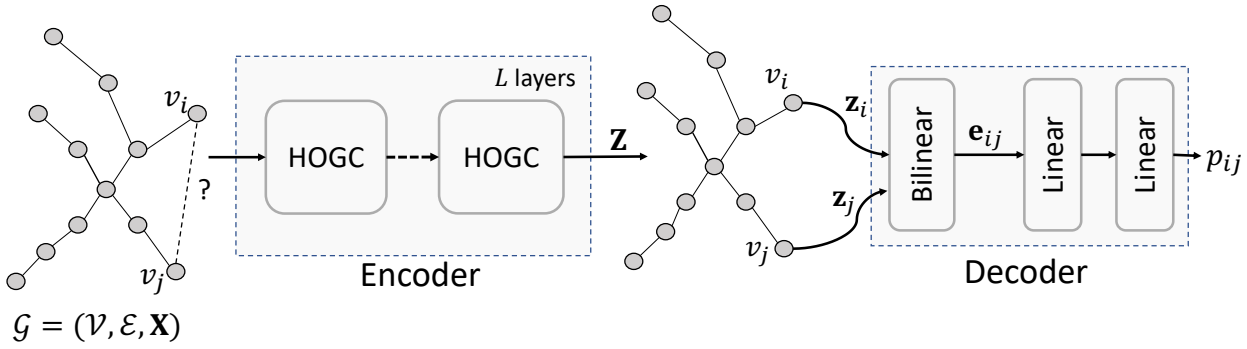


Figure 6.1: Block diagram of proposed HOGCN model with L HOGC layers. Given a biomedical interaction network $\mathcal{G}$ with initial features $\mathbf{X}$ for biomedical entities, the encoder mixes the feature representation of neighbors at various distances and learn final representation $\mathbf{Z}$. The decoder takes the representation $\mathbf{z}_i$ and $\mathbf{z}_j$ of nodes v_i and v_j to learn the representation $\mathbf{e}_{ij}$ for the edge (denoted by ?) and predict probability p_{ij} of its existence.

and aggregates incoming messages from its immediate neighbors. Representation for each biomedical entity can be obtained after L steps of message passing and feature aggregation. However, such message passing operation is limited to average pooling of features from immediate neighbors and thus is unable to learn feature differences among neighbors at various distances [28].

Neighborhood nodes at various distances provide network structure information at different granularities [34,130,131,132,133]. Taking k-hop neighborhoods into consideration, we aim at aggregating information from various distances at every message passing step. Different powers of adjacency matrices such as $\mathbf{A}^1, \mathbf{A}^2, \mathbf{A}^3, \ldots, \mathbf{A}^k$ provide information about the network structure at different scales. Higher-order message passing operations can therefore learn to mix their representations using various powers of the adjacency matrix at each message passing step.

6.3.2 Graph convolutional networks (GCNs)

Graph convolutional networks (GCNs) are the generalization of convolution operation from regular grids such as images or texts to graph structured data [46,134]. The key idea of GCNs is to learn the function to generate the node's representation by repeatedly aggregating information from immediate neighbors. The graph convolutional layer is defined as:

$$\mathbf{H}^{(l)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) \tag{6.1}$$

where $\mathbf{H}^{(l-1)}$ and $\mathbf{H}^{(l)}$ are the input and output activations, $\mathbf{W}^{(l)}$ is a trainable weight matrix of the layer l, σ is the element-wise activation, and $\hat{\mathbf{A}}$ is a symmetrically normalized adjacency matrix with self-connections $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_{|\mathcal{V}|})\mathbf{D}^{-\frac{1}{2}}$. A GCN model with L layers is then defined as:

$$\mathbf{H}^{(l)} = \begin{cases} \mathbf{X} & \text{if } l = 0 \\ \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) & \text{if } l \in [1, \ldots, L] \end{cases}$$

and $\mathbf{H}^{(L)}$ can be used to predict the probability of interactions between biomedical entities.

6.4 Higher-order graph convolution network

In this work, we develop a higher-order graph convolutional network (HOGCN) that takes an interaction network $\mathcal{G}$ as input and reconstruct the edges in the interaction network (Fig. 6.1). HOGCN has two main components:

- **Encoder**: a higher-order graph convolution encoder that operates on an interaction graph $\mathcal{G}$ and produces representations for biomedical entities by aggregating features from the neighborhood at various distances and

- **Decoder**: a bilinear decoder that relies on these representations to reconstruct the interactions in $\mathcal{G}$.

6.4.1 Higher-order graph encoder

We first describe the higher-order graph encoder, that operates on an undirected interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and learns the representations for biomedical entities.

We develop an encoder with higher-order Graph Convolution (HOGC) layer to mix feature representations from neighbors at k-distances. Specifically, HOGC layer considers the neighborhood information at different granularities and is defined as:

$$\mathbf{H}^{(l)} = \Big\|_{j \in \mathbb{P}} \sigma\left(\widehat{\mathbf{A}}^j \mathbf{H}^{(l-1)} \mathbf{W}_j^{(l)}\right) \tag{6.2}$$

where $\mathbb{P}$ is a set of integer adjacency powers, $\widehat{A}^j$ denotes the adjacency matrix $\widehat{A}$ multiplied j times, and $\|$ denotes column-wise concatenation [28]. Graph convolutional network [44] only considers the 1^{st} power of adjacency matrix and can be exactly recovered by setting $\mathbb{P} = \{1\}$ in Equation (6.2). Similarly, SkipGNN [19] considers direct and skip similarity and can be recovered by setting $\mathbb{P} = \{1, 2\}$ in Equation (6.2).

Fig. 6.2 shows a HOGC layer with $\mathbb{P} = \{0, 1, \ldots, k\}$ where k is maximum order of neighborhood considered in each HOGC layer. If $k = 0$, HOGC layer only considers the features of the biomedical entities and can capture the feature similarity between various biological entities. This is equivalent to a fully connected network with features of biomedical entities as input. For the HOGC layer, $\widehat{\mathbf{A}}^0$ is the identity matrix $\mathbf{I}_{|\mathcal{V}|}$ where $|\mathcal{V}|$ is the number of nodes in the network. This allows the HOGC layer to learn the transformation of node features separately and mix it with feature representations from neighbors.

The maximum order of neighborhood k and the number of trainable weight matrices $|\mathbb{P}|$, one per each adjacency power, can vary across layers. However, we set the same k for neighborhood aggregation and the same dimension d for all the weight matrices across all layers.

Neighborhood features from different adjacency powers $j \in \{0, 1, \ldots, k\}$ at layer $(l-1)$ are column-wise concatenated to obtain feature representation $\mathbf{H}^{(l-1)}$. As shown in Fig 6.2, weight $\mathbf{W}_{(\cdot)}^{(l)}$ at layer l can learn the arbitrary linear combination of the concatenated features to obtain $\mathbf{H}^{(l)}$. Specifically, the layer can assign different coefficients to different columns in the concatenated

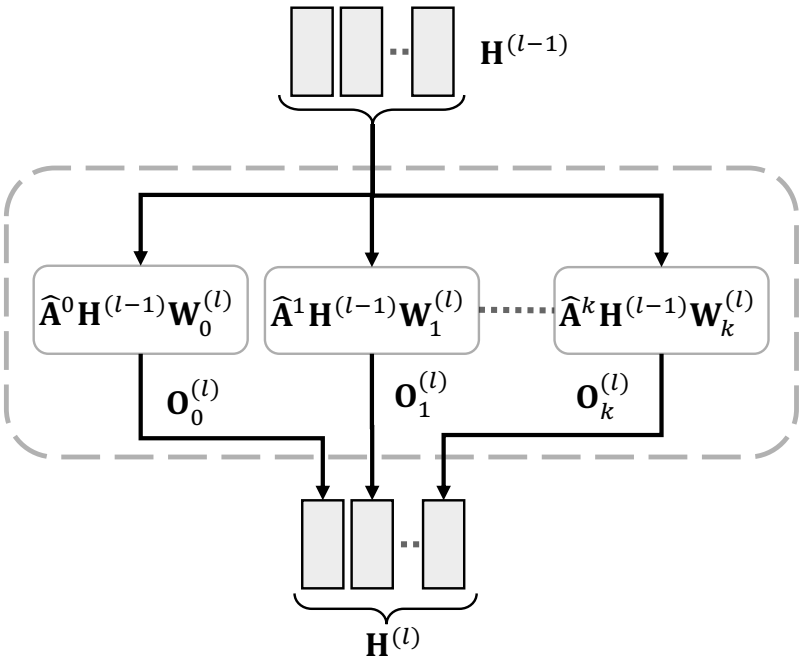


Figure 6.2: High-order graph convolution (HOGC) Layer with $\mathbb{P} = \{0, \ldots, k\}$. The feature representation $\mathbf{H}^{(l)}$ is a linear combination of the neighbors $\widehat{\mathbf{A}}^j\mathbf{H}^{(l-1)}$ at multiple distances j. $\mathbf{O}_j^{(l)}$ represents feature representation of neighbors at j distances for layer l.

matrix. For instance, the layer can assign positive coefficients to the columns produced by certain power of $\widehat{\mathbf{A}}$ and assign negative coefficients to other powers. This allows the model to learn feature differences among neighbors at various distances. We apply L HOGC layers to learn the latent representation $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d^*}$ for biomedical entities in the network, where $d^* = d \times |\mathbb{P}|$ and d is the dimension of node's representation for each adjacency power.

6.4.2 Interaction decoder

We introduced the encoder based on HOGC layers that learns feature representation $\mathbf{Z}$ for biomedical entities by mixing neighborhood information at multiple distances. Next, we discuss the decoder that reconstructs the interactions in $\mathcal{G}$ based on the representation $\mathbf{Z}$.

We adopt a bilinear layer to fuse the representation of biomedical entities v_i and v_j and learn the edge representation $\mathbf{e}_{ij}$. More precisely, we define a simple bilinear layer that takes the representa-

tion $\mathbf{z}_i$ and $\mathbf{z}_j$ as input:

$$\mathbf{e}_{ij} = \text{ELU}(\mathbf{z}_i^T \mathbf{W}_b \mathbf{z}_j + \mathbf{b}) \tag{6.3}$$

where $\mathbf{W}_b \in \mathbb{R}^{d \times d^* \times d^*}$ represents the learnable fusion matrix, $\mathbf{e}_{ij}$ is the representation of edge e_{ij} between nodes v_i and v_j, $\mathbf{b}$ denotes the bias of the bilinear layer. ELU is non-linearity.

The edge representation $\mathbf{e}_{ij}$ is then fed into 2-layered fully connected neural network to predict probability p_{ij} for edge e_{ij}:

$$p_{ij} = \text{sigmoid}(\text{FC}_2(\text{ELU}(\text{FC}_1(\mathbf{e}_{ij})))) \tag{6.4}$$

where $\text{FC}_1(\mathbf{e}_{ij}) = \mathbf{W}_1 \cdot \mathbf{e}_{ij} + \mathbf{b}_1$ denotes fully connected layer with weight $\mathbf{W}_1$ and bias $\mathbf{b}_1$, p_{ij} represents the probability that biomedical entities v_i and v_j interact.

So far, we have discussed the encoder and decoder of our proposed approach. Next, we describe the training procedure of our proposed HOGCN model. In particular, we explain how to optimize the trainable neural network weights in an end-to-end manner.

6.4.3 Training HOGCN

During HOGCN training, we employ binary cross entropy loss to optimize the model parameters

$$\mathcal{L}(v_i, v_j) = -A_{ij} \log(p_{ij}) - (1 - A_{ij}) \log(1 - p_{ij}) \tag{6.5}$$

and encourage the model to assign higher probability to observed interactions (v_i, v_j) than to randomly selected non-interactions. p_{ij} is the predicted interaction probability between v_i and v_j and A_{ij} denotes the ground-truth interaction label between these nodes. The final loss function considering all interactions is

$$\mathcal{L} = \sum_{(i,j) \in \mathcal{E}} \mathcal{L}(v_i, v_j) \tag{6.6}$$

We follow an end-to-end approach to jointly optimize over all trainable parameters and backpropagate the gradients through encoder and decoder of HOGCN.

Algorithm

HOGCN leverages biomedical network structure $\mathbf{A}$ along with additional information about biomedical entities as the initial feature representation $\mathbf{X}$. In this paper, we initialize the initial features $\mathbf{X}$ to be one-hot encoding *i.e.* $\mathbf{I}_{|\mathcal{V}|}$. The feature matrix $\mathbf{X}$ can be initialized with properties of biomedical entities or pre-trained embeddings from other network-based approaches such as DeepWalk, node2vec.

Algorithm 1 Training of HOGCN for biomedical interaction prediction

1: **Inputs:** $\hat{\mathbf{A}}$, $\mathbf{X}$, k
2: $\mathbf{H}^{(0)} = \mathbf{X}$
3: **for** $t = 1$ to T **do**
4: Sample mini-batch of training edges and their interaction labels
5: **for** $l = 1$ to L **do**
6: $\mathbf{B} := \mathbf{H}^{(l-1)}$
7: **for** $j = 1$ to k **do**
8: $\mathbf{B} := \hat{\mathbf{A}}\mathbf{B}$
9: $\mathbf{O}_j^{(l)} := \mathbf{B}\mathbf{W}_j^{(l)}$
10: $\mathbf{H}^{(l)} := \Big\Vert_{j \in \mathbb{P}} \mathbf{O}_j^{(l)}$
11: $\mathbf{Z} := \mathbf{H}^{(L)}$
12: $p_{ij} :=$ **Interaction Decoder**$(\mathbf{Z})$
13: Compute loss in (6.6)
14: Update model parameters via gradient descent

Given an adjacency matrix $\mathbf{A}$ and the initial node representations $\mathbf{X}$, higher-order neighborhood indicated by the higher power of the adjacency matrix is iteratively computed that makes the model more efficient. By adopting right-to-left multiplication, for instance, we can calculate $\hat{\mathbf{A}}^3\mathbf{H}^{(i)}$ as $\hat{\mathbf{A}}(\hat{\mathbf{A}}(\hat{\mathbf{A}}\mathbf{H}^{(i)}))$ (Line 8 in Algorithm 0). Representation $\mathbf{O}_j^{(l)}$ learned for the neighborhood at j distances are concatenated to obtain the representation $\mathbf{H}^{(l)}$ as shown in Fig. 6.2 (Line 11 in Algorithm 0). After passing through L HOGC layers, we obtain the final representation $\mathbf{Z}$ for biomedical entities. With the final representations $\mathbf{Z}$ and the mini-batch of training edges, we retrieve the embeddings for the nodes in training edges and feed them into the interaction decoder to compute their interaction probabilities.

The parameters of HOGCN are optimized with a binary cross-entropy loss (Equation (6.6)) in an end-to-end manner. Given two biomedical entities v_i and v_j, the trained model can predict the probability of their interactions.

6.5 Experimental design

We view the problem of biomedical interaction prediction as solving a link prediction task on an interaction network. We consider various interaction datasets and compare our proposed method with the state-of-the-art methods.

6.5.1 Datasets

We conduct interaction prediction experiments on four publicly-available biomedical network datasets:

- **BioSNAP-DTI** [135]: DTI network contains 15,139 drug-target interactions between 5,018 drugs and 2,325 proteins.
- **BioSNAP-DDI** [135]: DDI network contains 48,514 drug-drug interactions between 1,514 drugs extracted from drug labels and biomedical literature.
- **HuRI-PPI** [120]: HI-III human PPI network contains 5,604 proteins and 23,322 interactions generated by multple orthogonal high-throughput yeast two-hybrid screens.
- **DisGeNET-GDI** [136]: GDI network consists of 81,746 interactions between 9,413 genes and 10,370 diseases curated from GWAS studies, animal models and scientific literature.

Table 6.2 provides summary of datasets used in our experiments. Also, the table includes the average number of interactions for each biomedical entity which can be computed as $\frac{2|\mathcal{E}|}{|\mathcal{V}|}$.

Dataset	# Nodes	# Edges	Avg. node degree
DTI	5,018 drugs, 2,325 proteins	15,139	4.12
DDI	1,514 drugs	48,514	64.09
PPI	5,604 proteins	23,322	8.32
GDI	9,413 genes, 10,370 diseases	81,746	8.26

Table 6.2: Summary of the datasets used in our experiments.

6.5.2 Baselines

We compare our proposed model with the following network-based baselines for interaction prediction:

- Network similarity-based methods
  - **L3** [127] counts the number of paths with length-3 normalized by the degree for all the node pairs.
- Network embedding methods
  - **DeepWalk** [40] performs truncated random walk exploring the network neighborhood of nodes and applies skip-gram model to learn the d-dimensional embedding for each node in the network. Node features are concatenated to form edge representation and train a logistic regression classifier.
  - **node2vec** [15] extends DeepWalk by running biased random walk based on breadth/depth-first search to capture both local and global network structure.
- Graph convolution-based methods
  - **VGAE** [43] uses graph convolutional encoder with two GCN layers to learn representation for each node in the network and adopts inner product decoder to reconstruct adjacency matrix.
  - **GCN** [44] uses normalized adjacency matrix to learn node representations. The representation for nodes are concatenated to form feature representation for the edges and fully connected layer use these edge representation to reconstruct edges, similar to HOGCN. Setting $\mathbb{P} = \{1\}$ in our proposed HOGCN is equivalent to GCN.
  - **SkipGNN** [19] learns the node embeddings by combining direct and skip similarity between nodes. Setting $\mathbb{P} = \{1, 2\}$ in our proposed HOGCN is equivalent to SkipGNN.

6.5.3 Experimental setup

We split the interaction dataset into training, validation, and testing interactions in a ratio of 7:1:2 as shown in Table 6.2. Since the available interactions are positive samples, the negative samples are generated by randomly sampling from the complement set of positive examples. Five independent runs of the experiments with different random splits of the dataset are conducted to report the prediction performance. We use (1) area under the precision-recall curve (AUPRC) and (2) area under the receiver operating characteristics (AUROC) as the evaluation metrics. With these evaluation metrics, we expect positive interactions to have higher interaction probability compared to negative interactions. So, the higher value of AUPRC and AUROC indicates better performance.

We implement HOGCN using PyTorch [137] and perform all experiments on a single NVIDIA GeForce RTX 2080Ti GPU. We construct a 2-layered HOGC network with $k = 3$ for each layer. At each HOGC layer, the node mixes the feature representations from neighbors at distances $\mathbb{P} = \{0, 1, 2 \text{ and } 3\}$. The dimension of all weight matrices in HOGC layers is set to $d = 32$. All the weight matrices are initialized using Xavier initialization [114]. We train our model using mini-batch gradient descent with Adam optimizer [138] for a maximum of 50 epochs, with a fixed learning rate of 5×10^{-4}. We set the mini-batch size to 256 and the dropout probability [64] to 0.1 for all layers. Early stopping is adopted to stop training if validation performance does not improve for 10 epochs. The dimension of the edge feature $\mathbf{e_{ij}}$ from the bilinear layer is 64 followed by linear layers to project the edge features to edge probabilities. For baseline methods, we follow the same experimental settings discussed in [19].

6.6 Results

In this section, we investigate the performance and flexibility of HOGCN on interaction prediction using four different datasets. We further explore the robustness of HOGCN to sparse networks. Finally, we demonstrate the ability of HOGCN to make novel predictions with literature-based case studies.

6.6.1 Biomedical interaction prediction

We compare HOGCN against various baselines on biomedical interaction prediction tasks using four different types of interaction datasets including protein-protein interactions (PPIs), drug-target interactions (DTIs), drug-drug interactions (DDIs) and gene-disease associations (GDIs).

We randomly mask 20% of interactions from the network as a test set and 10% as a validation set. We train all models with 70% of interactions and evaluate their performances on test sets. The best set of hyperparameters is selected based on their performances on the validation dataset. Finally, the experiment is repeated for five independent random splits of the interaction dataset and the results with $\pm$ one standard deviation are reported in Table 6.3. All of our models used for the reported results are of same capacity (i.e. $\mathbb{P} = \{0, 1, 2, 3\}$ and $d = 32$).

Table 6.3 shows that HOGCN achieves huge improvement over network embedding methods such as DeepWalk and node2vec across all datasets. Specifically, HOGCN outperforms Deepwalk on

Dataset	Method	AUPRC	AUROC
DTI	DeepWalk	0.753 ± 0.008	0.735 ± 0.009
	node2vec	0.771 ± 0.005	0.720 ± 0.010
	L3	0.891 ± 0.004	0.793 ± 0.006
	VGAE	0.853 ± 0.010	0.800 ± 0.010
	GCN	0.904 ± 0.011	0.899 ± 0.010
	SkipGNN	0.928 ± 0.006	0.922 ± 0.004
	HOGCN	**0.937 ± 0.001**	**0.934 ± 0.001**
DDI	DeepWalk	0.698 ± 0.012	0.712 ± 0.009
	node2vec	0.801 ± 0.004	0.809 ± 0.002
	L3	0.860 ± 0.004	0.869 ± 0.003
	VGAE	0.844 ± 0.076	0.878 ± 0.008
	GCN	0.856 ± 0.005	0.875 ± 0.004
	SkipGNN	0.866 ± 0.006	0.886 ± 0.003
	HOGCN	**0.897 ± 0.003**	**0.911 ± 0.002**
PPI	DeepWalk	0.715 ± 0.008	0.706 ± 0.005
	node2vec	0.773 ± 0.010	0.766 ± 0.005
	L3	0.899 ± 0.003	0.861 ± 0.003
	VGAE	0.875 ± 0.004	0.844 ± 0.006
	GCN	0.909 ± 0.002	0.907 ± 0.006
	SkipGNN	0.921 ± 0.003	0.917 ± 0.004
	HOGCN	**0.930 ± 0.002**	**0.922 ± 0.001**
GDI	DeepWalk	0.827 ± 0.007	0.832 ± 0.003
	node2vec	0.828 ± 0.006	0.834 ± 0.003
	L3	0.899 ± 0.001	0.832 ± 0.001
	VGAE	0.902 ± 0.006	0.873 ± 0.009
	GCN	0.909 ± 0.002	0.906 ± 0.006
	SkipGNN	0.915 ± 0.003	0.912 ± 0.004
	HOGCN	**0.941 ± 0.001**	**0.936 ± 0.001**

Table 6.3: Average AUPRC and AUROC with ± one standard deviation on biomedical interaction prediction

AUPRC by 24.44% in DTI, 28.51% in DDI, 30.07% in PPI, and 13.79% in GDI. Although node2vec achieves better performance compared to DeepWalk by adopting a biased random walk, HOGCN still outperforms node2vec by a significant margin. DeepWalk and node2vec consider different orders of neighborhood defined by the window size and learns similar representations for the nodes

in that window. In contrast, HOGCN learns feature differences between neighbors at various distances to obtain feature representation for the node and thus achieves superior performance. The improved performance suggests that feature differences between different order neighbors provide important information for interaction prediction.

A network similarity-based method, L3 [127] outperforms network embedding methods across four datasets but is limited to a single aspect of network similarity i.e. the number of paths of length 3 connecting two nodes. So, L3 cannot be applied when other similarities between nodes such as similarity in features and common neighbors at various distances need to be considered. HOGCN overcomes these limitations and outperforms L3 across all interaction datasets with huge gain. In particular, HOGCN gains 3.5% AUPRC and 7.09% AUROC on PPI over L3 [127], which recently outperformed 20 network science methods in the PPI prediction problem.

Graph convolution-based methods such as GCN and VGAE achieves significant improvements over network embedding approaches but achieves comparable performance with L3. SkipGNN shows improvement over all other methods by incorporating skip similarity to aggregate information from second-order neighbors. Moreover, HOGCN with $k = 3$ achieves an improvement over all graph convolution-based methods. Specifically, HOGCN achieves improvement in AUPRC over VGAE [43] by 6.3%, GCN [44] by 4.8% and SkipGNN [19] by 3.6% on DDI dataset.

As HOGCN can learn the linear combination of node features at multiple distances, it can extract meaningful representations from the interaction networks. The results in Table 6.3 demonstrate that our approach with higher-order neighborhood mixing outperforms the state-of-the-art methods on real interaction datasets.

6.6.2 Exploration of HOGCN's drug representations

Next, we evaluate if HOGCN learns meaningful representation when feature representations of higher-order neighbors are aggregated. To this aim, we train GCN, SkipGNN, and HOGCN models on the DDI network to obtain the drug representations $\mathbf{Z}$. The learned drug representations are mapped to 2D space using t-SNE [86] and visualize them in Fig. 6.3.

Drugbank [73] provides information about drugs and their categories based on different characteristics such as involved metabolic enzymes, class of drugs, side effects of drugs, and the like. For this experiment, we collect drug categories from Drugbank and limit the selection of drug categories such that the training set doesn't contain any interactions between the drugs in the same category.

The selected drug categories are ACE Inhibitors and Diuretics (DBCAT002175), Penicillins (DBCAT000702), and Antineoplastic Agents (DBCAT000592) with 10, 24, and 16 drugs respectively. Although these drugs don't have direct interactions in the training set, we assume that these drugs share neighborhoods at various distances and can be explored accordingly with HOGCN.

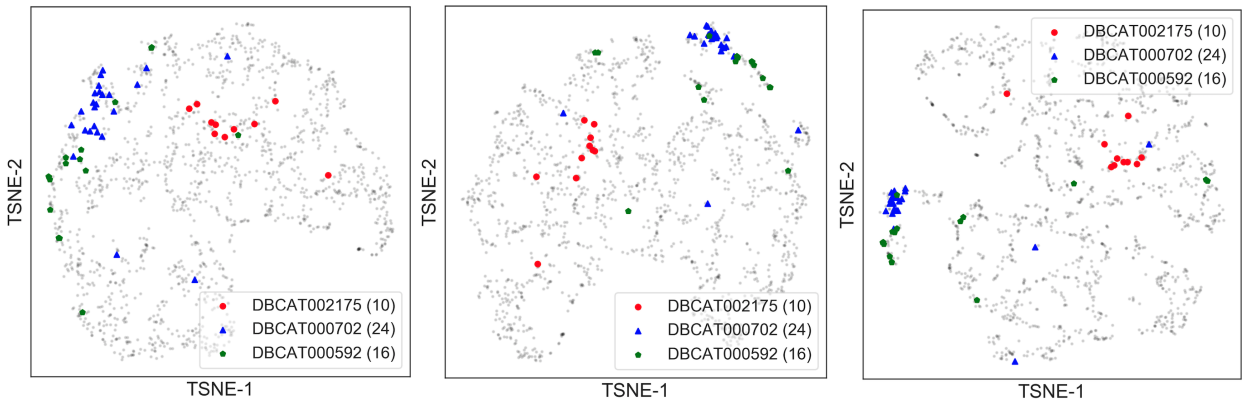


Figure 6.3: Visualization of learned representation for drugs with (a) GCN (b) SkipGNN (c) HOGCN. Drugs are mapped to the 2D space using t-SNE package with learned drug representations. Drugs categories such as DBCAT002175, DBCAT000702 and DBCAT000592 are highlighted. The number of drugs in each categroy is reported in legend. Best viewed on screen.

Fig. 6.3 shows the clustering structure in drugs' representations as neighborhood information at multiple distances are considered. Examining the figure, we observe that drugs in the same category are embedded close to each other in the 2D space when the model aggregates information from farther neighbors. For example, 24 drugs in the Penicillins (DBCAT000702) category (marked with blue triangles in Fig. 6.3) are scattered in the representation space learned by GCN that only considers feature aggregation from immediate neighbors (Fig. 6.3a). Note that these drugs don't have any direct interaction between themselves in the training set. Since GCN-based models can only average the representation from immediate neighbors, these drugs are mapped relatively farther to each other and closer to other interacting drugs. SkipGNN considers skip similarity to aggregate features from second-order neighbors and show relatively compact clusters compared to GCNs (Fig. 6.3b). On the other hand, HOGCN considers the higher-order neighborhood and learns similar representations for drugs that belong to the same category demonstrated by compact clustering structure in Fig. 6.3c even though no information about categorical similarity is provided to the model. This analysis demonstrates that HOGCN learns meaningful representation for drugs by aggregating feature representations from the neighborhood at various distances.

Next, we test if the clustering pattern in Fig. 6.3 holds across many drug categories. With this aim,

we consider all drug categories in DrugBank and compute the average Euclidean distance between each drug's representation and representations of other drugs within the same drug category. We then perform 2-sample Kolmogorov–Smirnov tests and found that HOGCN learns significantly more similar representations of drugs than expected by chance (p-value $= 4.93e-106$), GCNs (p-value $= 5.05e-56$) and SkipGNN ($1.29e-12$). Thus, this analysis indicates that HOGCN learns meaningful representations for drugs by aggregating neighborhood information at various distances.

6.6.3 Robustness to network sparsity

We next explore the robustness of network-based interaction prediction models to network sparsity. To this aim, we evaluate the performance with respect to the percentage of training edges varying from 10% to 70%. We make predictions on the rest of the interactions. We further use 10% of test edges for validation to select the best hyperparameter settings. For a fair comparison, we compare with graph convolution-based methods that aggregate information from direct and/or second-order neighbors.

Fig. 6.4 shows the robustness of HOGCN to network sparsity. HOGCN achieves strong performance in all tasks with different network sparsity. The performance of HOGCN steadily improves with the increase in training edges. The mixing of features from a higher-order neighborhood in HOGCN and SkipGNN shows improvement over GCN and VGAE that only consider direct neighbors. Since HOGCN can learn the linear combination of features from a 3-hop neighborhood for this experiment, it shows improvement over SkipGNN in almost all cases. This demonstrates that features from farther distances are informative for interaction prediction in sparse networks.

6.6.4 Calibrating model's prediction

All graph convolution-based model proposed for biomedical link prediction predicts the confidence estimate p_{ij} for interaction between two biomedical entities v_i and v_j. We thus test if a predicted confidence p_{ij} represents the likelihood of being true interaction. In other words, we expect the confidence estimate p_{ij} to be calibrated, *i.e.* p_{ij} represents true interaction probability [139]. For example, given 100 predictions, each with the confidence of 0.9, we expect that 90 interactions should be correctly classified as true interactions.

To evaluate the calibration performance, we use reliability diagrams [139] and Brier score [140]. In particular, the reliability diagram provides a visual representation of model calibration. These

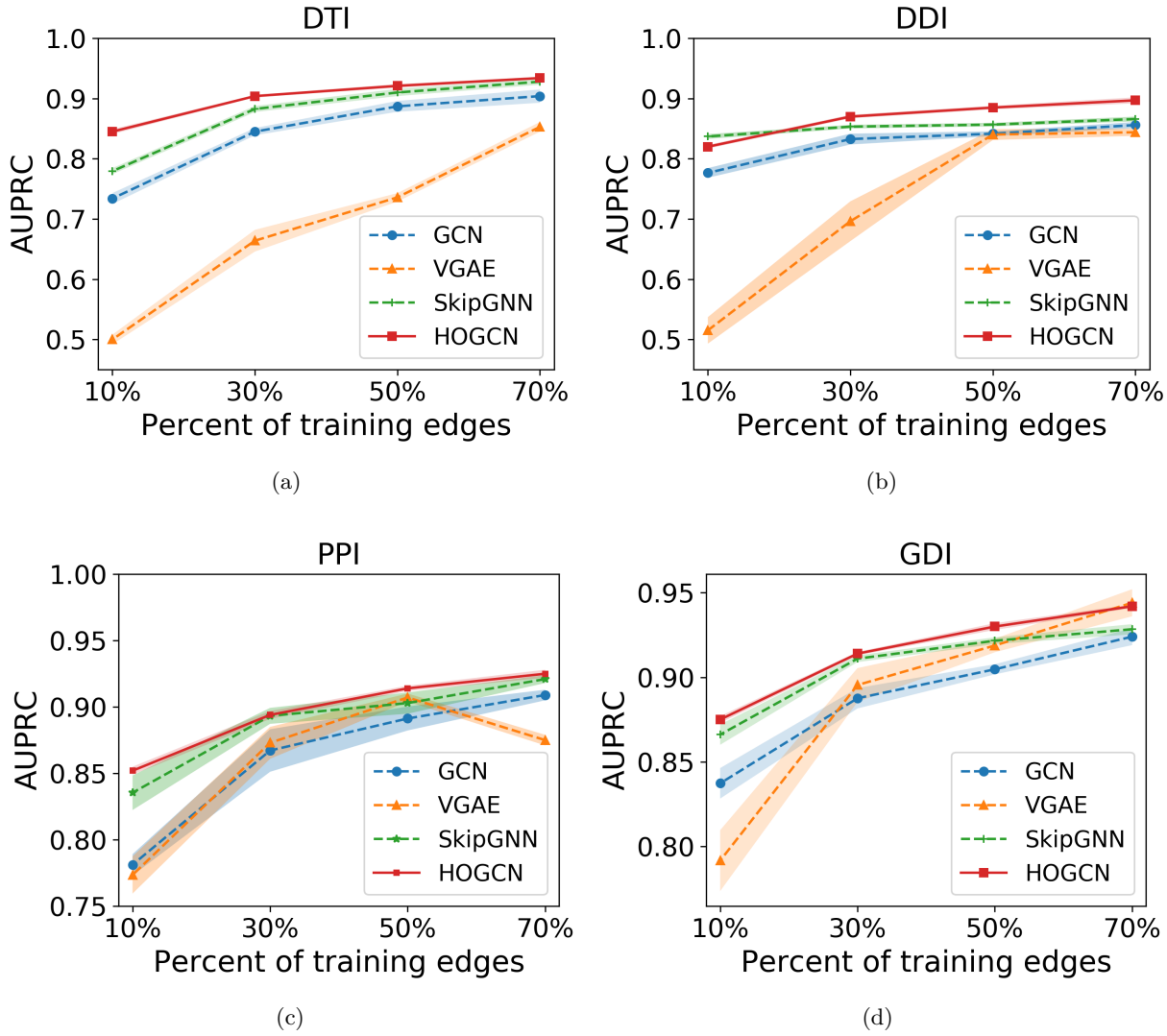


Figure 6.4: AUPRC comparison of HOGCN's performance with that of alternative approaches with respect to network sparsity. HOGCN consistently achieves better performance in various fraction of training edges.

diagrams plot the expected fraction of positives as a function of predicted probability [139]. A model with perfectly calibrated predictions is represented by a diagonal in Fig. 6.5. In addition to reliability diagrams, it is more convenient to have a scalar summary statistics of calibration. Brier score [140] is a proper scoring rule for measuring the accuracy of predicted probabilities. Lower Brier score indicates better calibration of a set of predictions. It is computed as the mean squared error of a predicted probability p_{ij} and the ground-truth interaction label A_{ij}. Mathematically, the

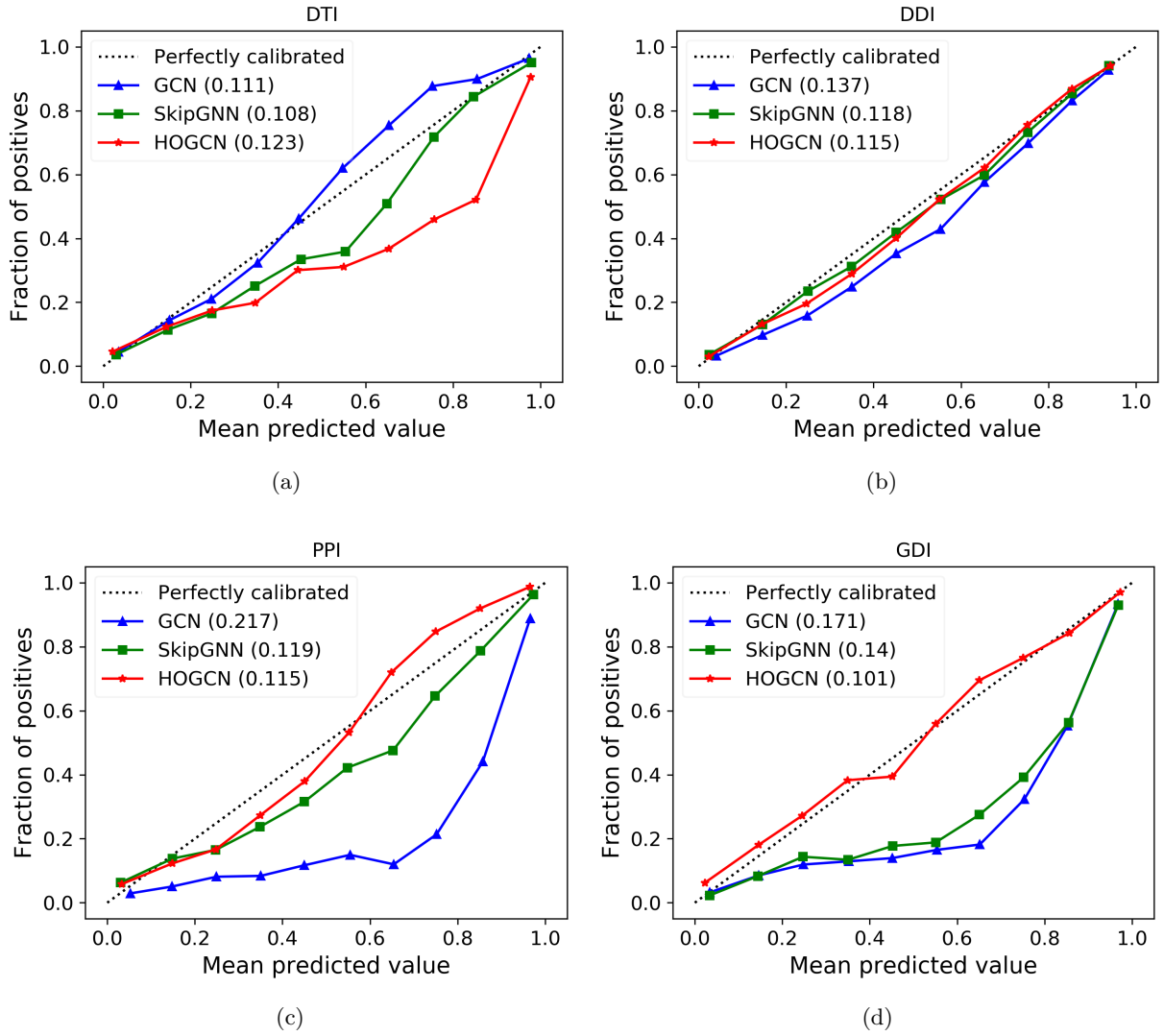


Figure 6.5: Reliability diagrams for different graph convolution-based methods. The calibration performance is evaluated with Brier score, reported in the legend (lower is better).

Brier score can be computed as:

$$\text{Brier score} = \frac{1}{|\mathcal{E}'|} \sum_{(i,j)=1}^{|\mathcal{E}'|} (p_{ij} - A_{ij})^2 \tag{6.7}$$

where $|\mathcal{E}'|$ denotes the number of test edges.

Fig. 6.5 shows the calibration plots for GCN, SkipGNN and HOGCN ($k = 3$). For DTI dataset,

SkipGNN show better calibration compared to GCN and HOGCN (Fig. 6.5a), indicating that second-order neighborhood information is appropriate and aggregating features from farther away makes model overconfident. For other datasets, GCNs are relatively overconfident for all predicted confidence. For example, approximately $20\% - 30\%$ of interactions are true positives among the interactions with high predicted confidence 0.8 in PPI (Fig. 6.5c) and GDI dataset (Fig. 6.5d). In contrast, HOGCN achieves a lower Brier score in comparison to the GCN and SkipGNN across DDI, PPI, and GDI datasets, alluding to the benefits of aggregating higher-order neighborhood features for calibrated prediction. This analysis demonstrates that HOGCN with higher-order neighborhood mixing makes accurate and calibrated predictions for biomedical interaction.

6.6.5 Impact of higher-order neighborhood mixing

In Section 6.6.3, we contrast HOGCN's performance with that of alternative graph convolution-based methods in varying fraction of edges. In this experiment, we aim to observe the performance of HOGCN when the order k is increased to allow the model to aggregate neighborhood information from farther away. We follow a similar setup as discussed in 6.6.3.

Fig. 6.6 shows the comparison of HOGCN with higher-order neighborhood mixing $k = \{3, 4, 5\}$. The prediction performance of HOGCN improves with the increase in the number of training interactions for all cases. The results show that HOGCN's performances are not sensitive to the hyperparameter settings of k for all datasets since for settings $k = \{3, 4, 5\}$, we achieve comparable performances across the datasets. This analysis indicates that the 3-hop neighborhood provides sufficient information for interaction prediction across all datasets and the performance remains stable even with a large value for k.

6.6.6 Investigation of novel predictions

Next, we perform the literature-based validation of novel predictions. Our goal is to evaluate the quality of HOGCN's novel predictions compared to that of GCN and SkipGNN and show that HOGCN predicts novel interactions with higher confidence. We consider GDIs and DDIs for this evaluation.

We first evaluate the potential of the HOGCN to make novel GDI predictions. We collect 1,134,942 GDIs and their scores from DisGeNET [136]. The score corresponds to the number and types of sources and the number of publications supporting the associations. With the score threshold of

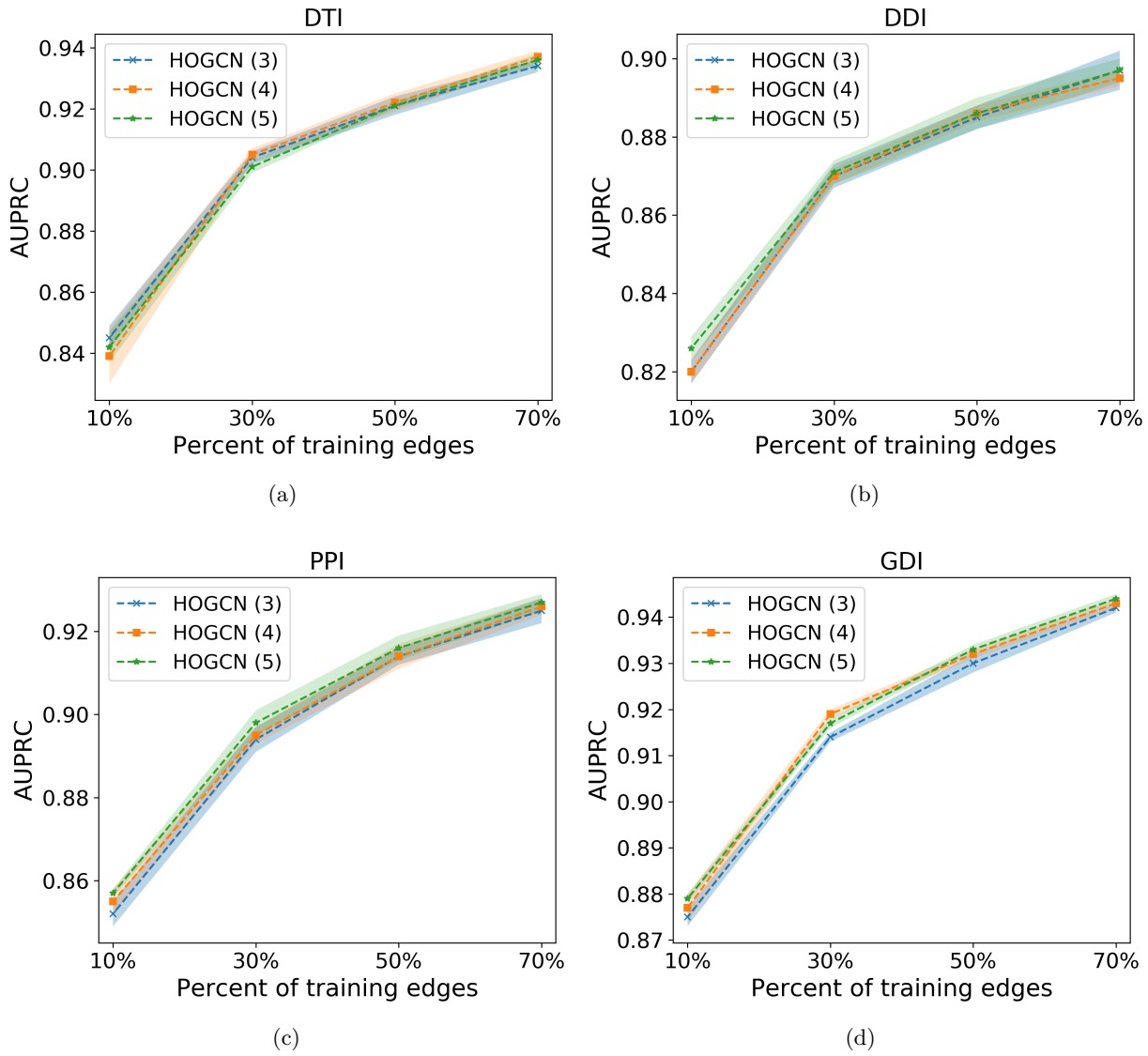


Figure 6.6: AUPRC comparison for higher-order message passing with different fractions of training edges. The values of k for different HOGCN models are reported in the legend.

0.18, we obtain 17,893 new GDIs that are not in the training set. We make predictions on these 17,893 GDIs with GCN, SkipGNN, and HOGCN ($k = 3$). Out of 17,893 GDIs, HOGCN predicts a higher probability than GCN for 17,356 (96.99%) GDIs and than SkipGNN for 11,418 (63.8%) GDIs. Table 6.4 shows the top 5 GDIs with a significant increase in interaction probabilities when higher-order neighborhood mixing is considered. We also provide the number of evidence from DisGenNet [136] to support these predictions. Improvement in predicted probabilities by HOGCN

models shows that aggregating feature representations from higher-order neighbors make HOGCN more confident about the potential interactions as discussed in Section 6.6.4.

Gene	Disease	Probability			No. of Evidence
		1	2	3	
PTGER1	Gastric ulcer	0.087	0.519	0.721	1
ANGPT1	Gastric ulcer	0.173	0.583	0.657	2
ABO	Pancreatic carcinoma	0.265	0.615	0.770	26
VCAM1	Endotoxemia	0.294	0.529	0.639	5
GPC3	Hepatoblastoma	0.307	0.540	0.598	17

Table 6.4: Novel prediction of GDIs with the number of evidence from DisGenNet supporting the interaction. GCN, SkipGNN and HOGCN are denoted by $1, 2$ and 3 respectively.

We select two predicted GDIs with a large number of supporting evidence and investigate the reason for the improvement in predicted confidence with HOGCN. Specifically, we choose gene-disease pairs (a) ABO and Pancreatic carcinoma (26 pieces of evidence) and (b) GPC3 and Hepatoblastoma ((17 pieces of evidence). To explain the prediction, the subnetworks containing all shortest paths between these pairs are selected. In particular, there are 49 shortest paths of length 3 between ABO and Pancreatic carcinoma including 6 diseases and 15 genes (Fig. 6.7a). Similarly, there are 20 shortest paths of length 3 between GPC3 and Hepatoblastoma including 6 diseases and 9 genes (Fig. 6.7b). Since these nodes are 3-hop away from each other and GCNs can only consider immediate neighbors, GCNs assign low confidence to these interactions.

Examining the subnetwork in Fig. 6.7a, we found that most of the diseases are related to a cancerous tumor in the pancreas and the prostate. Furthermore, pancreatic carcinoma is associated with other diseases such as Pancreatic neoplasm, malignant neoplasm of pancreas, and malignant neoplasm of prostate [136]. Since ABO is linked with diseases that are related to pancreatic carcinoma and other genes are related to these diseases as well, HOGCN captures such association (Fig 6.7a) even though they are farther away in the network. Similarly, HOGCN predicts association for GPC3 and Hepatoblastoma (Fig. 6.7b).

Next, we perform a similar case study for DDIs and evaluate the predictions against DrugBank [73]. For this experiment, we make a prediction for every drug pair with GCN, SkipGNN and HOGCN and exclude the interactions that are already in the training set. Table 6.5 shows the top 5 interactions with an increase in interaction probabilities when highers orders of the neighborhood are considered. As discussed in Section 6.6.4, HOGCN makes predictions with higher confidence compared to GCN and SkipGNN for the interactions that are likely to be a true positive.

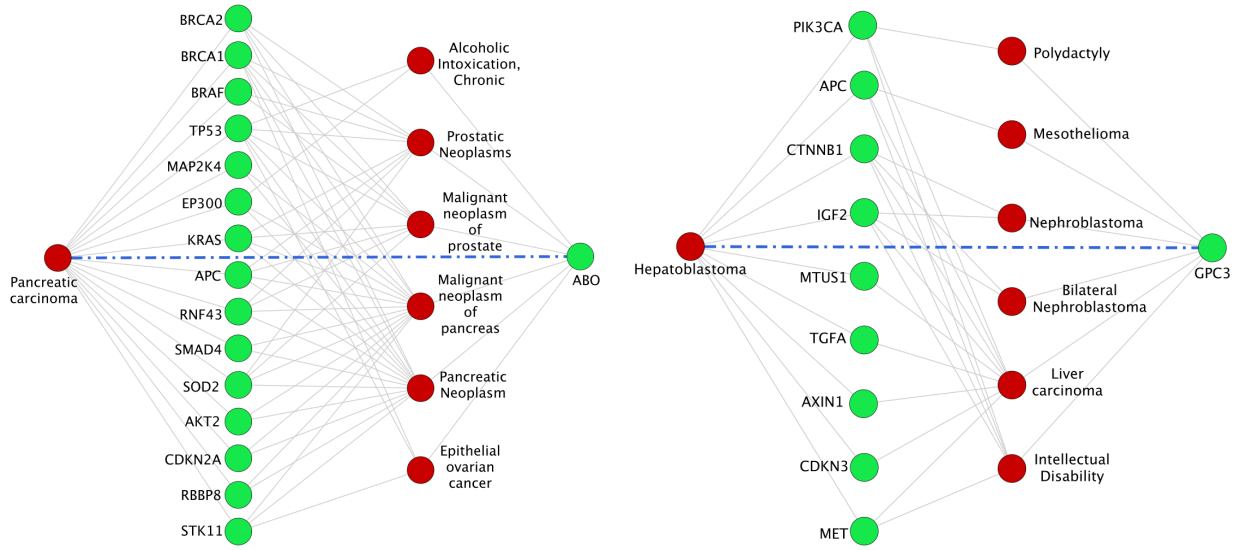


Figure 6.7: Subnetwork with predicted interactions (marked by bold dashed lines) between (a) ABO and Pancreatic carcinoma (b) GPC3 and Hepatoblastoma and all shortest paths between these pairs. The known interactions are presented as gray lines. Diseases are presented as dark circles and genes are presented as white circles.

Drug 1	Drug 2	Probability			Evidence
		1	2	3	
Nelfinavir	Acenocoumarol	0.192	0.318	0.417	[141]
Praziquantel	Itraconazole	0.609	0.721	0.811	[142]
Cisapride	Droperidol	0.618	0.725	0.823	[143]
Dapsone	Warfarin	0.632	0.720	0.885	[144]
Levofloxacin	Tobramycin	0.663	0.760	0.823	[145]

Table 6.5: Novel prediction of DDIs with the literature evidence supporting the interaction. GCN, SkipGNN and HOGCN are denoted by $1, 2$ and 3 respectively.

Moreover, we validate the false positive DDI predictions of GCNs and investigate the subnetwork for these drugs in DDI networks to reason the predictions. Table 6.6 shows the top 5 interactions with a significant decrease in predicted confidence compared to GCN-based models. Since these DDIs are false positives [73], GCN-based models make overconfident predictions for such DDIs. In contrast, HOGCN significantly reduces the predicted confidence for these DDIs to be true positive, indicating that the higher-order neighborhood allows HOGCN to identify false positive predictions. In particular, HOGCN can identity false positive DDI between Belimumab and Estazolam even though they are 3-hop away from each other.

Drug 1	Drug 2	Probability with k		
		1	2	3
Tranylcypromine	Melphalan	0.925	0.478	0.065
Belimumab	Estazolam	0.912	0.477	0.178
Methotrimeprazine	Cloxacillin	0.907	0.406	0.065
Hydrocodone	Melphalan	0.905	0.193	0.012
Ibrutinib	Mecamylamine	0.899	0.398	0.353

Table 6.6: Predicted probability for negative DDIs. GCN, SkipGNN and HOGCN are denoted by $1, 2$ and 3 respectively.

We select subnetwork involving the drugs to investigate the reason for such predictions. Fig. 6.8 shows the subnetwork with all shortest paths between the drugs in Table 6.6. Examining the figure, we observe that the drugs in these false positive DDIs have common immediate neighbors for all cases. GCN makes wrong predictions for these DDIs with high confidence. However, SkipGNN becomes less confident about the interaction being true positive by considering the skip similarity. HOGCN further reduces the predicted confidence for Tranylcypromine and Melphalan to 0.065, indicating that there is no association between these drugs.

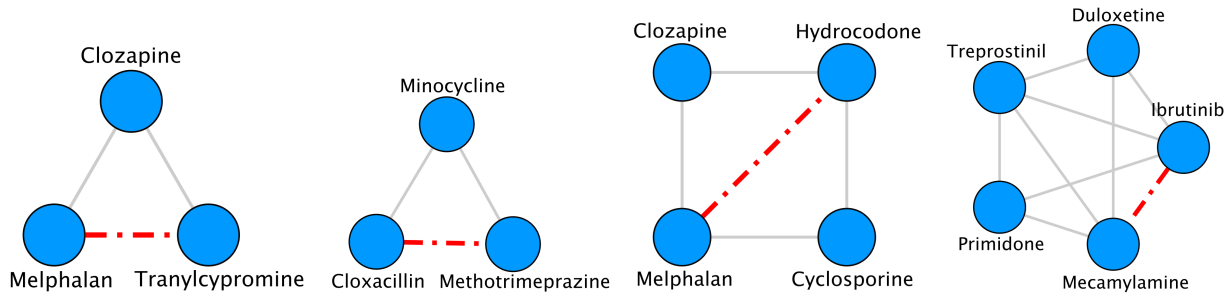


Figure 6.8: Subnetwork containing false positive predictions (marked by dark dashed lines) and all shortest paths between (a) Tranylcypromine and Melphalan (b) Methotrimeprazine and Cloxacillin (c) Hydrocodone and Melphalan and (d) Ibrutinib and Mecamylamine. Other known interactions are presented as gray gray lines. Dark circles denotes drugs.

These case studies show that HOGCN with higher-order neighborhood mixing not only provide information for the identification of novel interactions but also help HOGCN to reduce false positive predictions.

6.7 Conclusion

We present a novel deep graph convolutional network (HOGCN) for biomedical interaction prediction. Our proposed model adopts a higher-order graph convolutional layer to learn to mix the feature representation of neighbors at various scales. Experimental results on four interaction datasets demonstrate the superior and robust performance of the proposed model. Furthermore, we show that HOGCN makes accurate and calibrated predictions by considering higher-order neighborhood information.

There are several directions for future study. Our approach only considers the known interactions to flag potential interactions. There are other sources of biomedical information such as various physicochemical and biological properties of biomedical entities that can provide additional information about the interaction and we plan to investigate the integration of such features $\mathbf{X}$ into the model. As HOGCN aggregates the neighborhood information at various distances and can flag novel interactions, it would be interesting to provide interpretable explanations for the predictions in the form of a small subgraph of the input interaction network $\mathcal{G}$ that are most influential for the predictions [146, 147].

Part II

Probabilistic Model Selection for Network Representation Learning

Chapter 7

Joint inference for neural network depth and neuron activations

In part I, we designed and developed neural network architectures to learn representation for biological entities and demonstrated the effectiveness of these networks in predicting novel biological interactions. These neural network architectures are crucial in learning the feature representation from data. However, the design of such complex networks relies heavily on the prior's knowledge and experience. This process is not only time consuming and error-prone, but also may lead to complex models that are prone to overfitting. Specifically, finding the optimal number for hidden layers (depth) involves computationally expensive and time-consuming search process such as grid search or random search. Moreover, various structure selection methods [2,49,50] and regularization techniques [48,64] are used which requires the hyperparameter to be set carefully to determine the best configuration for a given dataset and avoid overfitting.

To address this problem, we develop a Bayesian model selection method to jointly infer the most plausible network depth warranted by data and perform dropout regularization simultaneously. In particular, to infer network depth we define a beta process over the number of hidden layers which allows it to go to infinity. Layer-wise activation probabilities induced by the beta process modulate neuron activation via binary vectors of a conjugate Bernoulli process. Experiments across domains show that by adapting network depth and dropout regularization to data, our method achieves superior performance comparing to state-of-the-art methods with well-calibrated uncertainty estimates.

7.1 Introduction

Both dropout regularization and network depth are critical to the success of deep neural networks (DNNs) [3, 7, 148, 149]. Finely tuned or selected structures empower DNNs with proper model capacity that can not only efficiently capture statistical regularities in data but also avoid being deceived by random noise into "discovering" non-existent relationships.

Dropout as an effective regularization method sparsifies a neural network's structure by randomly deleting neurons along with their connections in training to prevent it from overfitting [47, 150]. Recent studies extend the method from a Bayesian perspective to allow automatic tuning of the dropout probability in large models [1, 52, 53, 54, 55, 149]. Although dropout and its variants can effectively govern model capacity, without uncertainty calibration deep models tend to be overly confident with their predictions [7, 151]. Probabilistic inference approaches for network structure selection propose to bypass or down-weight certain hidden layers to reduce training cost and build smaller models that perform just as well as larger ones [4, 5, 6, 7]. These methods lead to more efficient or better-calibrated models only by reducing the depth of a pre-determined network structure. They cannot scale the network structure up to accommodate incrementally available data beyond the upper limit of its capacity.

We thus propose a novel Bayesian model selection framework to jointly infer network depth and perform dropout regularization simultaneously. In particular, we model the depth of a network structure as a stochastic process by defining the beta process [152, 153] over the number of hidden layers to enable it to go to infinity in theory, as in Figure 7.1a (left). The beta process induces layer-wise activation probabilities, which allows its conjugate Bernoulli process to generate a binary vector per hidden layer to prune neurons for regularization, as in Figure 7.1b. The probabilistic inference framework automatically balances network depth and dropout regularization by computing a marginal likelihood over the hidden layers and their neuron activations, and provides well-calibrated uncertainty estimates for predictions. The exact computation of the marginal likelihood is intractable due to the nonlinear nature of neural networks with a potentially infinite number of hidden layers. We thus employ the structured stochastic variational inference [154, 155] with a continuous relaxation on the Bernoulli variables to efficiently approximate the integral with a continuum of lower bounds. The relaxation allows to maintain differentiability and mitigate overfitting via model averaging of the sampled hidden layers and neuron activations. Benefiting from theoretical analysis, we readily integrate the joint inference with the parameter learning process through an alternating approach to efficiently update the parameters, and perform hidden-layer sampling and dropout regularization.

We analyze the behavior of our joint inference framework over multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), and evaluate their performance across domains. The experiments show that our method leads to a compact neural network by balancing its depth and dropout regularization with uncertainty calibration, and achieves superior performance comparing to state-of-the-art dropout and structure selection methods. We also demonstrate our method on a continual learning task. By enabling both network depth and neuron activations to dynamically evolve to accommodate incrementally available data, we can alleviate catastrophic forgetting.

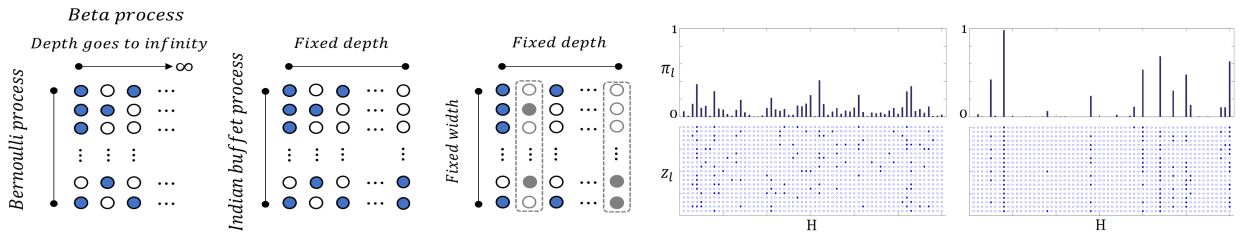


Figure 7.1: (a) Demonstrations of our proposed framework (left), a dropout variant (middle), and typical structure selection methods (right). Our framework enables depth goes to infinity by modeling the number of hidden layers with a beta process. The dropout variant defines an Indian buffet process per layer to infer width, but depth is fixed [1, 2]. Structure selection methods can only reduce depth of a pre-determined network structure [3, 4, 5, 6]. (b) On top, random draws from two beta processes with $\alpha > \beta$ on the left and $\beta > \alpha$ on the right over hidden layer function space $\mathbf{H} = \{\mathbf{h}_l \mid l \to \infty\}$. An atom location is $\delta_{\mathbf{h}_l}$ indexing a hidden layer function $\mathbf{h}_l$, and the height denotes its activation probability π_l. For both cases, the conjugate Bernoulli processes at bottom are obtained by random filled dots $z_{ml} = 1$ with probability π_l and empty dots $z_{ml} = 0$ with probability $1 - \pi_l$. So each column is a binary vector $\mathbf{z}_l$ corresponding to layer l.

7.2 Related work

Dropout is an effective neural network regularization technique [47, 48]. By randomly pruning neurons and their connections from a network structure with a dropout rate during training, it can prevent DNNs from overfitting [49, 50, 51]. [52] interprets dropout from a Bayesian perspective to quantify uncertainty. Variational dropout proposes a stochastic gradient variational inference approach to learn the dropout rate from data with a constraint on large dropout rate values [53]. [54] extends the variational dropout to set unbounded individual dropout rate per weight. Concrete dropout proposes a continuous relaxation of the dropout's discrete masks to automatically tune the dropout probability, and obtain its uncertainty estimates [55]. Bayesian nonparametrics are also applied to extend the variational dropout approaches by defining an Indian buffet process (IBP)

over neurons per hidden layer [1] or over channels in CNNs [2] to infer network widths, as in Figure 7.1a (middle). Another body of work leverage the non-parametric Bayesian framework to infer the dimensionality of latent representation encoded by DNNs [56, 57].

Structure selection methods improve training efficiency by reducing the depth of a pre-arranged network structure, as in Figure 7.1a (right). Some lead to well-performed smaller networks, and others achieve better uncertainty calibration. In order to build smaller models that perform just as well, [3] proposes to reduce a large neural network by merging adjacent hidden layers with only linear relationships being activated in between. A stochastic depth method randomly drops a subset of hidden layers from a large network by bypassing them with an identity function in each mini-batch to speed up the training session, but it still deploys the large network structure at test time [4]. [6] shows that the multiplicative noise from dropout induces structured shrinkage priors over a neural network's weights, and they further extend the shrinkage framework based on ResNet structures [58] to model the probabilities of hidden layers being used. In particular, the work also indicates that the framework is equivalent to the stochastic depth regularization [4]. [5] proposes to tune a hidden layer's influence on a prediction by learning a bypass variable between adjacent-layer connections and skip connections with a variational Bayes method. The method also prunes neurons separately. [7] proposes a Bayesian model averaging method to down-weight deeper layers by directly connecting every hidden layer to the output layer, and achieve competitive performances with uncertainty calibration. [59] achieves model effectiveness and computational efficiency by inferring a distribution of connections and units in the context of local winner-takes-all DNNs with IBP.

7.3 The joint inference framework

We propose to model network depth as a stochastic process, and jointly perform dropout regularization upon the inferred hidden layers. Theoretical analysis shows that by approximating its marginal likelihood the inference framework achieves an optimal balance between network depth and neuron activations as Bayesian model selection.

7.3.1 Network structure with infinite hidden layers

Let $\mathbf{h}_l$ denote the l-th hidden layer composed of neurons (i.e., non-linear activation functions) $\sigma(\cdot)$. The neural network has the form:

$$\mathbf{h}_l = \sigma(\mathbf{W}_l \mathbf{h}_{l-1}) \bigotimes \mathbf{z}_l + \mathbf{h}_{l-1} \qquad l \in \{1, 2, ..., \infty\} \tag{7.1}$$

where $\mathbf{W}_l \in \mathrm{R}^{M \times M}$ is the layer l's weight matrix with a Gaussian prior $p(\mathbf{W}) = \mathcal{N}(\mathbf{W}|0, s_w^2\mathbf{I})$, $\bigotimes$ denotes element-wise multiplication of two vectors, and M is the maximum number of neurons in a layer. For simplicity, we set M to be the same for all hidden layers. In particular, we prune the l-th layer's outputs by multiplying them elementwisely by a binary vector $\mathbf{z}_l$ where its element variable $z_{ml} \in \{0, 1\}$. Each random variable z_{ml} takes the value 1 with $\pi_l \in [0, 1]$ indicating activation probability of the l-th layer, as in Figure 7.1b. The combination of the previous layer's outputs with the current layer's via skip connections not only avoids vanishing gradient but also propagates the output of the last hidden layer with activated neurons directly to the output layer $f(\cdot)$.[1]

Given a dataset $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ with N input-output pairs $(\mathbf{x}_n, \mathbf{y}_n)$ as training examples, for a regression task we express the likelihood of the neural network as:

$$p(D|\mathbf{Z}, \mathbf{W}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n | f(\mathbf{x}_n; \mathbf{Z}, \mathbf{W}), s^2\mathbf{I}) \tag{7.2}$$

where $\mathbf{Z}$ is a binary matrix whose l-th column is $\mathbf{z}_l$, and $\mathbf{W} = \{\mathbf{W}_l\}$ denotes the set of the weight matrices. s^2 denotes likelihood noise with $\mathbf{I}$ as an identity matrix with the same dimensionality as the training examples' outputs. For classification tasks, we replace $f(\cdot)$ with a softmax function and the normal distribution with a multinoulli distribution.

7.3.2 Beta process over layer number

We treat the number of hidden layer functions $\mathbf{h}_l$ as a stochastic process by defining a beta process over their space $\mathbf{H} = \{\mathbf{h}_l\}_{l=1}^{\infty}$, as shown in Figure 7.1b.

Conceptually, we generate a beta process B from its base measure B_0 as $B \sim \mathrm{BP}(c, B_0)$ by drawing a set of samples $(\mathbf{h}_l, \pi_l) \in \mathbf{H} \times [0, 1]$ from a non-homogeneous Poisson process [152]. Let $B = \sum_l \pi_l \delta_{\mathbf{h}_l}$, where $\delta_{\mathbf{h}_l}$ is a unit point mass at $\mathbf{h}_l$. In our setting, a pair $(\mathbf{h}_l, \pi_l)$ corresponds to a hidden layer

[1] A graphical demonstration of the network structure is in Appendix.

function $\mathbf{h}_l \in \mathbf{H}$ and its activation probability $\pi_l \in [0, 1]$. We define a conjugate Bernoulli process $\mathbf{Z}_{m\cdot} \sim \mathrm{BeP}(B)$ as $\mathbf{Z}_{m\cdot} = \sum_l z_{ml}\delta_{\mathbf{h}_l}$ at the same locations $\delta_{\mathbf{h}_l}$ as B where z_{ml} are independent Bernoulli variables with a probability that $z_{ml} = 1$ equals to π_l. As in (7.1), $z_{ml} = 1$ activates the m'th neuron in layer l.

Computationally, we employ the stick-breaking construction of beta-Bernoulli processes [153, 156] as

$$z_{ml} \sim \mathrm{Ber}(\pi_l), \quad \pi_l = \prod_{j=1}^{l} \nu_j, \quad \nu_l \sim \mathrm{Beta}(\alpha, \beta) \tag{7.3}$$

where ν_l are sequentially drawn from a beta distribution. According to the formulation, π_l is decreasing with l. α and β are the hyperparameters governing preference over a balance between network depth and dropout regularization. In particular, if $\alpha > \beta > 1$, the hidden layers tend to have lower activation probabilities with less number of activated neurons, as in Figure 7.1b left. The setting thus prefers a narrower but deeper network structure. On the other hand, $\beta > \alpha > 1$ favors shallower but wider network structure, where fewer hidden layers tend to have higher activation probabilities with more neurons being activated, as in Figure 7.1b right.

We thus define a prior over $\mathbf{Z}$ via the beta process as

$$p(\mathbf{Z}, \boldsymbol{\nu}|\alpha, \beta) = p(\boldsymbol{\nu}|\alpha, \beta)p(\mathbf{Z}|\boldsymbol{\nu}) = \prod_{l=1}^{\infty} \mathrm{Beta}(\nu_l|\alpha, \beta) \prod_{m=1}^{M} \mathrm{Ber}(z_{ml}|\pi_l) \tag{7.4}$$

7.3.3 Marginal likelihood over network structures

By combining the beta process prior over hidden layers in (7.4) with the neural network structure in (7.2), we can infer the number of hidden layers L by integrating over $\mathbf{Z} = \{\mathbf{z}_l\}_{l=1}^{L}$ and $\boldsymbol{\nu} = \{\nu_l\}_{l=1}^{L}$:

$$p(D|\mathbf{W}, L, \alpha, \beta) = \int p(D|\mathbf{Z}, \mathbf{W})p(\mathbf{Z}, \boldsymbol{\nu}|\alpha, \beta)d\mathbf{Z}d\boldsymbol{\nu} \tag{7.5}$$

The exact computation of this marginal likelihood is intractable due to the non-linearity of the neural network and $L \to \infty$.

7.3.4 Efficient inference with SSVI

We employ structured stochastic variational inference (SSVI) to approximate the marginal likelihood with a variational lower bound (ELBO) retaining the dependence between network structures and hyperparameters [154, 155, 157]. In particular, we define the variational distribution as

$$q(\mathbf{Z}, \boldsymbol{\nu}|\{a_k\}_{k=1}^K, \{b_k\}_{k=1}^K) = q(\boldsymbol{\nu})q(\mathbf{Z}|\boldsymbol{\nu}) = \prod_{k=1}^{K} \text{Beta}(\nu_k|a_k, b_k) \prod_{m=1}^{M} \text{ConBer}(z_{mk}|\pi_k) \tag{7.6}$$

where $\pi_k = \prod_{j=1}^{k} \nu_j$, and $\{a_k, b_k\}_{k=1}^K$ are variational parameters of $q(\boldsymbol{\nu})$, and K denotes a truncation level, which can be relaxed as in [158]. We relax the constraint of the discrete variables with continuous ones by reparameterizing the Bernoulli distribution into a concrete Bernoulli distribution [159, 160]:

$$\text{ConBer}(z_{mk}|\pi_k) = \tau \frac{\pi_k (z_{mk})^{-\tau-1}(1-\pi_k)(1-z_{mk})^{-\tau-1}}{(\pi_k (z_{mk})^{-\tau} + (1-\pi_k)(1-z_{mk})^{-\tau})^2} \tag{7.7}$$

where τ is a temperature controlling the distribution smoothness. We generate random samples from the distribution by first sampling from a logistic distribution as the external source of randomness and then putting the samples through a logistic function as follow:

$$z_{mk} = \frac{1}{1 + \exp(-\tau^{-1}(\log \pi_k - \log(1-\pi_k) + \epsilon))} \qquad \epsilon \sim \text{Logistic}(0, 1) \tag{7.8}$$

This reparameterization of the Bernoulli distribution allows us to backpropagate the gradients of the lower bound with respect to the parameters while sampling from it.

We derive a lower bound of the log marginal likelihood in (7.5) as:

$$\log p(D|\mathbf{W}, L, \alpha, \beta) \geq \int q(\mathbf{Z}, \boldsymbol{\nu}) \log p(D|\mathbf{Z}, \mathbf{W}) d\mathbf{Z} d\boldsymbol{\nu} + \int q(\mathbf{Z}, \boldsymbol{\nu}) \log \frac{p(\mathbf{Z}, \boldsymbol{\nu})}{q(\mathbf{Z}, \boldsymbol{\nu})} d\mathbf{Z} d\boldsymbol{\nu} \tag{7.9}$$

The ELBO from the right-hand side of (7.9) becomes our objective function:

$$\mathcal{L}(\mathbf{Z}, \boldsymbol{\nu}, \mathbf{W}) = \mathbf{E}_{q(\mathbf{Z},\boldsymbol{\nu})}[\log p(D|\mathbf{Z}, \mathbf{W})] - \text{KL}[q(\mathbf{Z}|\boldsymbol{\nu})||p(\mathbf{Z}|\boldsymbol{\nu})] - \text{KL}[q(\boldsymbol{\nu})||p(\boldsymbol{\nu})] \tag{7.10}$$

The first term of the ELBO in (7.10) is the averaged log-likelihood. The second and the third terms are the KL divergence between the variational distribution and the true prior, respectively. In particular, for the second term we relax the prior $p(\mathbf{Z}|\boldsymbol{\nu})$ with a concrete Bernoulli as in (7.7) with different temperatures. For the third term the KL divergence between the variational distribution

and the prior distribution over $\boldsymbol{\nu}$ is:

$$\begin{aligned}\mathrm{KL}[q(\boldsymbol{\nu})||p(\boldsymbol{\nu})] = \sum_k \ln\left(\frac{\mathrm{B}(\alpha,\beta)}{\mathrm{B}(a_k,b_k)}\right) + (a_k-\alpha)\,\psi(a_k) \\ + (b_k-\beta)\,\psi(b_k) + (\alpha-a_k+\beta-b_k)\,\psi(a_k+b_k)\end{aligned} \tag{7.11}$$

where $\psi(\cdot)$ denotes a di-gamma function.

Theorem 1. *Assume $\hat{\mathbf{W}}$ is the optimized value of the variable, and the likelihood and the induced prior on $\Pi_{\mathbf{Z}}$ for $\mathbf{Z}$ are specified in (7.2) and (7.4), in the large sample limit, the limiting of the ELBO in (7.10) becomes equivalent to the Bayesian information criterion [161], such that $BIC(\mathbf{Z}) = \log p(D|\mathbf{Z},\hat{\mathbf{W}}) - \frac{|\mathbf{Z}|}{2}\log\frac{N}{2\pi}$, where $|\mathbf{Z}|$ denotes the number of activated neurons in the network structure.*

To prove Theorem 1, let the binary matrix $\mathbf{Z}$ index the activated neurons with its non-zero elements, and we denote their weights as $\mathbf{W}_{\mathbf{Z}}$. We employ a Gaussian approximation to the posterior distribution as follows. Let

$$p(\mathbf{W}_{\mathbf{Z}}|D) = \frac{1}{p(D)}p(\mathbf{W}_{\mathbf{Z}},D) = \frac{1}{C}e^{-E(\mathbf{W}_{\mathbf{Z}})} \tag{7.12}$$

where $E(\mathbf{W}_{\mathbf{Z}})$ is equal to the negative log of the unnormalized log posterior:

$$E(\mathbf{W}_{\mathbf{Z}}) = -\log p(\mathbf{W}_{\mathbf{Z}},D) \tag{7.13}$$

with $C = p(D)$ being the normalization constant. We expand $E(\mathbf{W}_{\mathbf{Z}})$ around the mode $\hat{\mathbf{W}}_{\mathbf{Z}}$ with the Taylor series as

$$\begin{aligned}E(\mathbf{W}_{\mathbf{Z}}) \approx E(\hat{\mathbf{W}}_{\mathbf{Z}}) + (\mathbf{W}_{\mathbf{Z}}-\hat{\mathbf{W}}_{\mathbf{Z}})^T g \\ + \frac{1}{2}(\mathbf{W}_{\mathbf{Z}}-\hat{\mathbf{W}}_{\mathbf{Z}})^T H(\mathbf{W}_{\mathbf{Z}}-\hat{\mathbf{W}}_{\mathbf{Z}})\end{aligned} \tag{7.14}$$

where g is the gradient and H is the Hessian of the energy function evaluated at the mode:

$$g = \nabla E(\mathbf{W}_{\mathbf{Z}})|_{\hat{\mathbf{W}}_{\mathbf{Z}}} \qquad H = \frac{\partial^2 E(\mathbf{W}_{\mathbf{Z}})}{\partial \mathbf{W}_{\mathbf{Z}}\partial \mathbf{W}_{\mathbf{Z}}^T}|_{\hat{\mathbf{W}}_{\mathbf{Z}}} \tag{7.15}$$

The second term in (7.14) can be dropped, since the gradient term $(\mathbf{W}_{\mathbf{Z}}-\hat{\mathbf{W}}_{\mathbf{Z}})^T g = 0$ due to the

mode $\hat{\mathbf{W}}_{\mathbf{Z}}$. We thus have

$$\begin{aligned}
& p(\mathbf{W}_{\mathbf{Z}}|D) \\
&= \frac{1}{p(D)} p(\mathbf{W}_{\mathbf{Z}}, D) \\
&= \frac{1}{C} e^{-E(\mathbf{W}_{\mathbf{Z}})} \\
&\approx \frac{1}{C} e^{-E(\hat{\mathbf{W}}_{\mathbf{Z}})-0-\frac{1}{2}(\mathbf{W}_{\mathbf{Z}}-\hat{\mathbf{W}}_{\mathbf{Z}})^T H(\mathbf{W}_{\mathbf{Z}}-\hat{\mathbf{W}}_{\mathbf{Z}})} \\
&= \frac{e^{-E(\hat{\mathbf{W}}_{\mathbf{Z}})}}{C} \exp[-\frac{1}{2}(\mathbf{W}_{\mathbf{Z}} - \hat{\mathbf{W}}_{\mathbf{Z}})^T H(\mathbf{W}_{\mathbf{Z}} - \hat{\mathbf{W}}_{\mathbf{Z}})] \\
&= N(\mathbf{W}_{\mathbf{Z}}|\hat{\mathbf{W}}_{\mathbf{Z}}, H^{-1}) \\
C &= p(D) \\
&= \int p(\mathbf{W}_{\mathbf{Z}}, D) d\mathbf{W}_{\mathbf{Z}} \\
&= e^{-E(\hat{\mathbf{W}}_{\mathbf{Z}})} (2\pi)^{\frac{|\mathbf{Z}|}{2}} |H|^{-\frac{1}{2}}
\end{aligned} \tag{7.16}$$

We abuse the notation a little by using $|\mathbf{Z}|$ to denote the number of non-zero elements.

The log marginal likelihood can thus be approximated with the Gaussian approximation as follows:

$$\begin{aligned}
\log p(D) &\approx -E(\hat{\mathbf{W}}_{\mathbf{Z}}) - \frac{1}{2}\log|H| + \frac{|\mathbf{Z}|}{2}\log 2\pi \\
&= \log p(D|\hat{\mathbf{W}}_{\mathbf{Z}}) + \log p(\hat{\mathbf{W}}_{\mathbf{Z}}) - \frac{1}{2}\log|H| + \frac{|\mathbf{Z}|}{2}\log 2\pi
\end{aligned} \tag{7.17}$$

The penalization terms following $\log p(D|\hat{\mathbf{W}}_{\mathbf{Z}})$ are a measure of model complexity. The second term can be ignored by assuming a uniform prior. By approximating each H_i by a fixed matrix $\hat{H}$ in the third term, $H = \sum_{i=1}^{N} H_i$, where $H_i = \nabla\nabla \log p(D_i|\mathbf{W}_{\mathbf{Z}})$, we have

$$\begin{aligned}
\log|H| &= \log|N\hat{H}| \\
&= \log(N^{|\mathbf{Z}|}|\hat{H}|) \\
&= |\mathbf{Z}|\log N + \log|\hat{H}|
\end{aligned} \tag{7.18}$$

where we assume H is a full rank matrix. $\log|\hat{H}|$ term can be ignored due to its independence of N. We thus have the Bayesian information criteria (BIC) score:

$$\log p(D) \approx \log p(D|\hat{\mathbf{W}}_{\mathbf{Z}}) - \frac{|\mathbf{Z}|}{2}\log\frac{N}{2\pi} \tag{7.19}$$

Optimizing the ELBO produces the best approximation to the true posterior within the space of

distributions, as well as the tightest lower bound on the true marginal likelihood. Our variational inference framework optimizes a lower bound to this BIC with respect to $\mathbf{Z}$ in order to have the most plausible network structure size.

This theorem indicates that optimizing the limiting case of our SSVI framework can therefore be seen as optimizing the popular model selection criteria. We adopt the implicit differentiation of beta distribution for backpropagation [162], which allows us to stochastically optimize $\mathcal{L}$ with respect to the variational parameters every time we sample a network structure. We iteratively maximize ELBO with respect to the variational parameters and weights $\mathbf{W}$, and approximate the ELBO using random samples from the variational distribution.

We approximate the posterior predictive distribution by sampling from the variational posterior distribution for predictions. Thus, the predictive distribution for new data $\mathbf{x}^*$ is

$$p(\mathbf{y}^*|\mathbf{x}^*, \hat{\mathbf{W}}, \{\hat{a}_k\}, \{\hat{b}_k\}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{Z}, \hat{\mathbf{W}}) q(\mathbf{Z}, \boldsymbol{\nu}|\{\hat{a}_k\}, \{\hat{b}_k\}) d\mathbf{Z} d\boldsymbol{\nu} \tag{7.20}$$

where $\hat{\mathbf{W}}$ is the MAP estimate of the neural network weights based on mini-batch gradient descent.

7.4 Experiments

We analyze the behavior of our framework by applying it to MLPs and CNNs on a variety of tasks. We study how it enables network depth to adapt to dataset sizes, and capture different types of uncertainty on a synthetic dataset. Our framework outperforms state-of-the-art methods on the UCI datasets with uncertainty estimation. To obtain performance comparison, we perform Bayesian Optimisation and Hyperband (BOHB) to determine the best configurations for each method [163]. We also investigate the effect of truncation level K and M, and show that their settings have no influence on the performance as long as they are reasonably large. The experiments with image datasets show that our framework improves CNN performances with more compact network structures and well-calibrated predictive distributions. Lastly, we demonstrate our method in two continual learning tasks, and show that by allowing network depth to dynamically augment to accommodate incrementally available information we can effectively alleviate catastrophic forgetting. [2]

[2] Implementation details of all the experiments are in Appendix.

7.4.1 Synthetic experiments

We analyze our method's behaviors given different dataset sizes, and its uncertainty estimation. We incrementally generate 20, 500 and 2000 data points from a periodic function [5]:

$$y = \sin(6x) + 0.4x^2 - 0.1x^3 - x\cos(9\sqrt{\exp(x)}) + \epsilon \tag{7.21}$$

where $\epsilon \sim \mathcal{N}(0, 0.1^2)$ with 20% for validation. The data are shown in Figure 7.2a bottom row. We set the maximum number of neurons per layer $M = 20$, and use leaky ReLU activation functions in the input and hidden layers with batch normalization to retain stability. We simulate 5 samples per mini-batch to approximate the ELBO in (7.10), and evaluate convergence by computing the cross-correlations of the sample values over 3000 epochs.

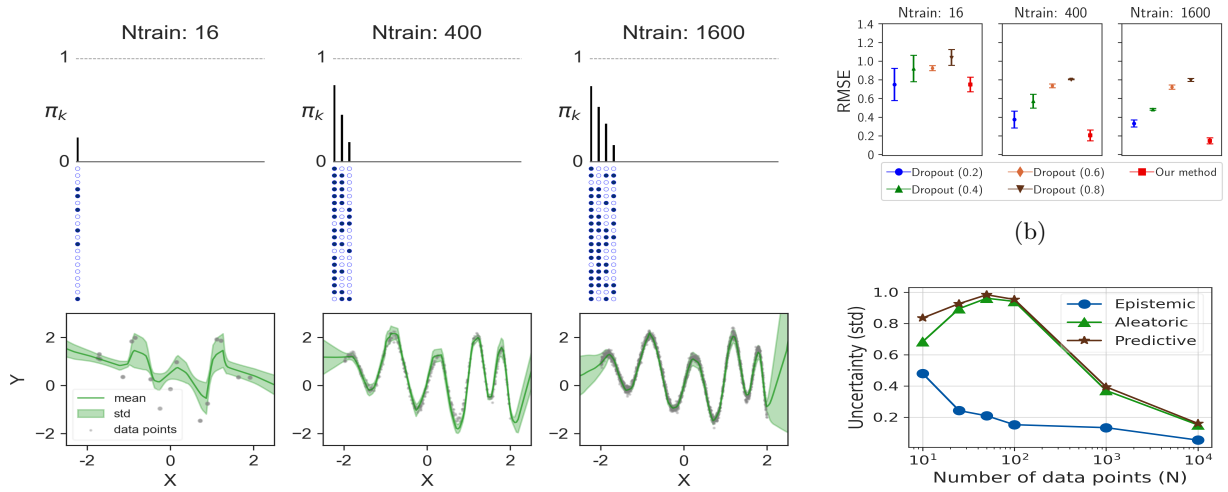


Figure 7.2: (a) Top row shows the activation probabilities π_k (black bars) of the inferred hidden layers, and the neuron activations $\mathbf{Z}$ (filled dots denote activated neurons). The network becomes deeper with more activated neurons as the training dataset size increases. Bottom row shows the predictive distributions overlaying the data points, and the green bands are ±one standard deviation over the predictions of 5 sampled network structures. (b) Predictive performance evaluation of our method and vanilla dropout with different dropout rates for the three cases. (c) Our method's estimates of different uncertainties as the number of data points increases.

As shown in Figure 7.2a, to accommodate more information from larger datasets, the neural network becomes deeper with more activated neurons. Although we set the truncation level $K = 100$ for all the training cases, only the first few hidden layers have activated neurons (i.e., 1 layer, 3 layers, and 4 layers for the respective cases). Figure 7.2a bottom row shows how the predictive distributions

in (7.20) are capturing the true function and estimating uncertainty as the dataset size increases. We compare our method's performance with vanilla dropout [47] whose backbone structure sizes are the same as the respective ones inferred by our method. The predictive performances on 700 held-out data points in Figure 7.2b indicate that by obtaining the critical balance between network depth and neuron activations our method achieves more stable performance with less variance on the small dataset, and significantly better performance on the larger ones.

The datasets generated with a known variance allow us to assess the different types of uncertainty of our method's predictions. In particular, the epistemic uncertainty is obtained by drawing multiple samples of the inferred depth and neuron activation, and evaluating them on the test set of 700 data points from the same data distribution. Figure 7.2c shows that the epistemic uncertainty decreases as the amount of data points increases. The aleatoric uncertainty approaching the true uncertainty (0.1) shows an increasingly improved estimate as more data is given. Our method's predictive uncertainty obtained by combining epistemic and aleatoric uncertainties converges to a constant value.

7.4.2 Performance comparisons on UCI datasets

We evaluate all methods on UCI datasets [164] using standard splits [165], and report their performance in terms of log-likelihood (LL) and root mean square error (RMSE). In particular, we compare the performances with vanilla dropout (Dropout), concrete dropout (CD) [55], Indian buffet process dropout (IBPD) [1], Bayesian architecture learning (BAL) [5], stochastic depth (SDepth) [4], depth uncertainty networks (DUNs) [7], and Deep Ensembles (DeepEns) [166]. We perform hyperparameter optimization with Bayesian Optimisation and Hyperband (BOHB) to determine the best configurations including backbone-structure depths and hyperparameter settings for each method [163], as suggested in [7]. The detailed implementation is in Appendix.

In Figure 7.3, we rank the methods from 1 to 8 based on their mean performance across each dataset and metric, and report mean ranks with $\pm$ one standard deviation. Our joint inference framework outperforms other methods in terms of both test log-likelihood (LL) and RMSE by achieving the best rank across all datasets. In particular, LL measures both accuracy and uncertainty calibration [167]. The narrower standard deviations of our method indicate its robustness. The vanilla dropout achieves the second-best rank followed closely by the Deep Ensembles in terms of LL and concrete dropout in terms of RMSE.

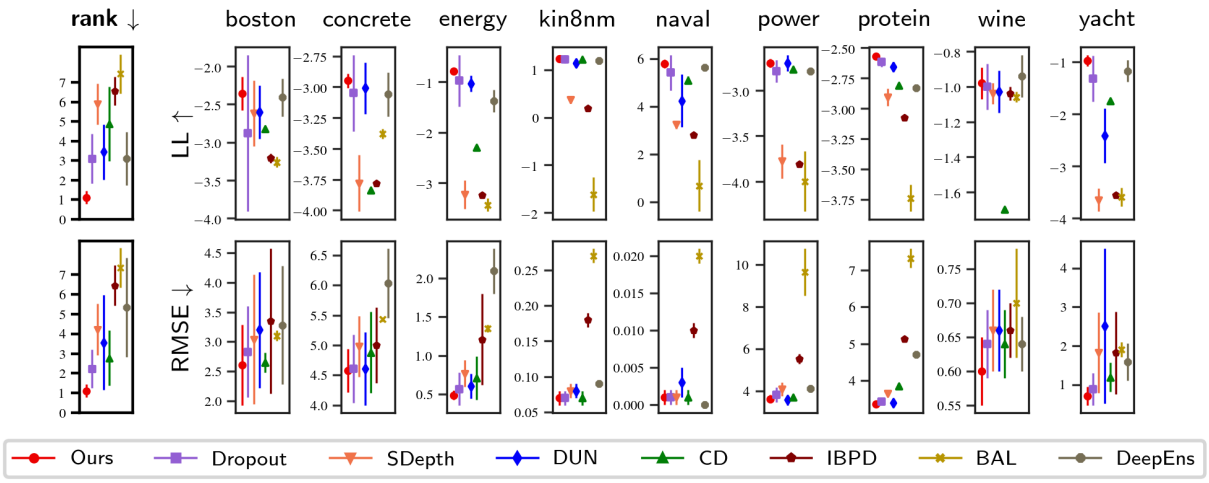


Figure 7.3: Mean values with ± one standard deviation for test log-likelihood (LL) and RMSE on UCI standard splits. Average ranks are computed across datasets. For LL, higher is better. For rank and RMSE, lower is better. The metrics are defined as in [7] with the codes generating the plots. Details are in Appendix.

7.4.3 Effect of truncation level

We further investigate the effect of truncation level K on our method's performance, and compare it with the backbone-structure depth L of dropout variants and structure selection methods.

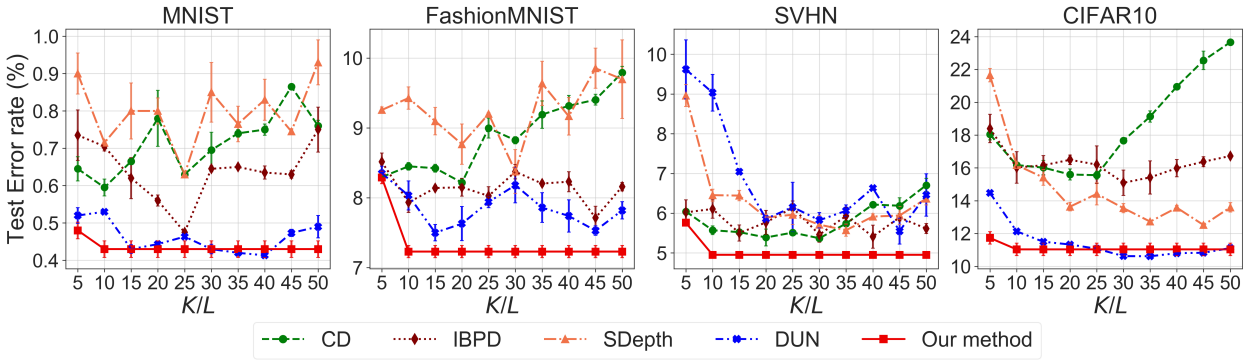
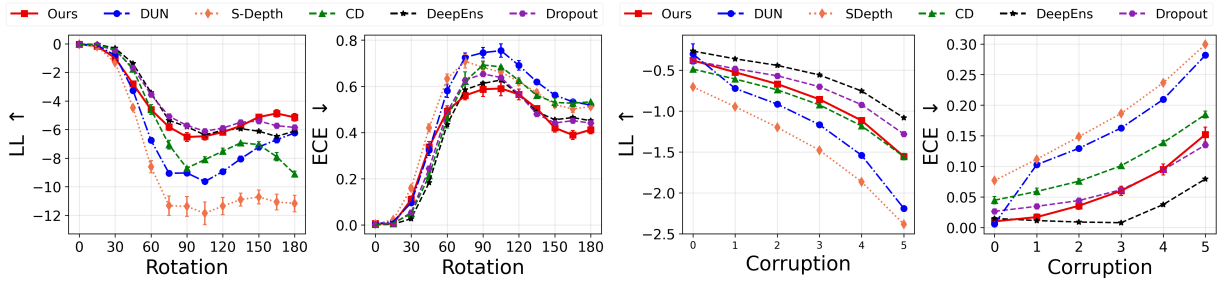


Figure 7.4: Analysis of the influence of our method's truncation level K and other methods' backbone-structure depth L in CNNs with the four image datasets. As long as K is reasonably large (≥ 10), it no longer has an influence on our method's performances. On the contrary, the performance of the other methods depends on L. As L becomes large, they tend to have an overfitting problem with increased test errors. By jointly inferring network depth and neuron activations, our method is robust to overfitting, and achieve the best performances.

All the methods have the same maximum width $M = 64$ (i.e., the maximum number of feature maps in convolutional layers). In Figure 7.4, we apply all the methods on CNNs with the backbone-structure depth L and the truncation level K over the range $L = K \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$ for classification tasks of four image datasets: MNIST [168], FashionMNIST [169], SVHN [170], and CIFAR10 [171]. When $L = K = 5$, all the methods have an underfitting problem due to the limited model capacity, although our method still achieves competitive or better performances. As the truncation level becomes reasonably large (≥ 10), it no longer affects our method's performance. The results are consistent with **Theorem 1**, as our joint inference essentially works as Bayesian model selection on a network structure truncation. On the contrary, the depths of backbone structure L affect the other methods to a great extent. The increase of the test errors shows that they all have an overfitting problem when the backbone-structure depths become large. Individual method reaches its own best performance, as L increases. Our method outperforms them for all these cases with orders of magnitude less computing resources across the four datasets. The respective evaluations in Figure 7.6 (i.e., $M = 64$) show that our method only activates neurons in 6, 6, 10 and 6 hidden layers for predictions on MNIST, FashionMNIST, SVHN, and CIFAR10, respectively. Only 10% of total neurons in the truncation level are activated for all the cases. This suggests by jointly inferring network depth and neuron activations our method is quite robust to overfitting, and the balance between the two also leads to superior performances with compact network structures. In contrast, the backbone-structure sizes of the dropout variants and structure selection methods have to be set carefully to determine the best configuration for a given dataset.



(a) Rotated MNIST

(b) Corrupted CIFAR10

Figure 7.5: Evaluation of uncertainty estimates for (a) MNIST for varying degrees of rotation and (b) CIFAR10 at varying degrees of corruption severities with log-likelihood (LL) and expected calibration error (ECE), as in [7].

We also evaluate the uncertainty calibration of our framework and other methods with their best settings in Figure 7.4. In Figure 7.5a, we train all methods on MNIST and evaluate their predictive distributions on increasingly rotated digits [172]. The uncertainty calibration of our method is more

robust to the data with large angle rotations than other methods, and it achieves the performance with the lowest expected calibration error (ECE) [173]. In Figure 7.5b, to make a fair and rigorous comparison as in [7], we also evaluate the corruption robustness of all methods on the CIFAR10 dataset with 16 types of algorithmically generated corruptions [174]. Each type of corruption has 5 levels of severity. The Deep Ensembles outperform other methods, and our model achieves comparable performance at all corruption levels.

7.4.4 Effect of M

We next assess the influence of the maximum number of neurons/feature maps (M) on our method with classification tasks on the four image datasets. Figure 7.6 shows the evolution of the number of inferred convolutional layers as M increases. When $M = 16$, we tend to have deeper network structures to compensate for the relatively narrow layers. As M increases, the structures become shallower. When M is reasonably large, it has no influence on the inferred structure depth. The percentage of activated neurons in the truncation level remains relatively stable across different values of M. This suggests that our method can adapt both the network depth and the neuron activations to maintain the best performances as M changes.

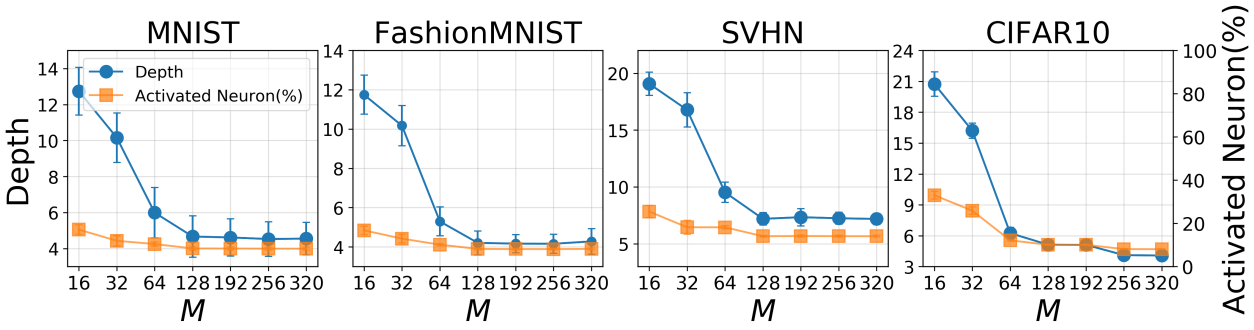


Figure 7.6: Influence of the maximum number of feature maps (M) on our method with four image datasets. When M is small, we tend to have deeper network structures (blue). As M becomes reasonably large (e.g., ≥ 128), it tends not to have influence on the inference of network structure sizes. Meanwhile, the percentages of activated neurons in the truncation level are stable (orange).

7.4.5 Case study on Continual learning

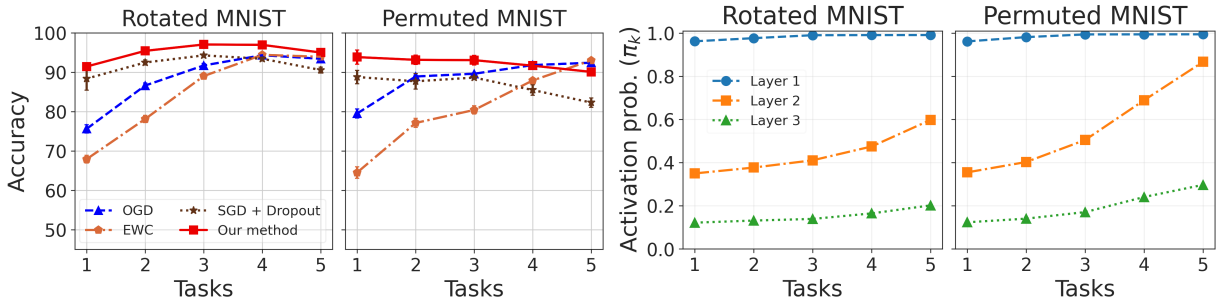
Continual learning is an important application since real-world tasks are dynamic and non-stationary. Machine learning models need to learn consecutive tasks without forgetting how to perform previously trained ones. Assuming the pre-determined backbone structures are sufficient to accommo-

date all information from the continual tasks, network regularization approaches alleviate catastrophic forgetting by regularizing the updates of neural weights [175]. Dropout and its variants are applied to learn an implicit gating mechanism that activates different gates for different tasks [176,177]. However, the rigid backbone structure constrains the applicability of these methods in real-world settings.

We slightly modify our method to enable network depth and neuron activations to dynamically evolve to accommodate incrementally available data. Given a set of sequentially arriving datasets $\{D_t\}$ where each may contain a single datum, we update our ELBO in (7.10) as:

$$\mathcal{L}_t = \mathbf{E}_{q_t(\mathbf{Z},\boldsymbol{\nu})}[\log p(D_t|\mathbf{Z},\boldsymbol{\nu},\mathbf{W})] - \mathrm{KL}[q_t(\mathbf{Z}|\boldsymbol{\nu})||q_{t-1}(\mathbf{Z}|\boldsymbol{\nu})] - \mathrm{KL}[q_t(\boldsymbol{\nu})||q_{t-1}(\boldsymbol{\nu})] \tag{7.22}$$

by replacing the priors $p(\mathbf{Z}|\nu)$ and $p(\nu)$ with their variational approximation for the previous dataset $q_{t-1}(\mathbf{Z}|\nu)$ and $q_{t-1}(\nu)$. We initialize them as $q_0(\mathbf{Z}|\nu) = p(\mathbf{Z}|\nu)$ and $q_0(\nu) = p(\nu)$. The variational distribution in (7.6) after seeing the t-th dataset is recursively updated by taking the distribution after seeing the $(t-1)$-th dataset, multiplying by the likelihood and re-normalizing.



(a) (b)

Figure 7.7: (a) The validation accuracy averaged over five runs for each task after training on all tasks in sequence. (b) Evolution of layer-activation probabilities (π_k) with the five tasks.

We compare the performance of our method with baseline continual learning methods including elastic weight consolidation (EWC) [175], orthogonal gradient descent (OGD) [178], and stochastic gradient descent with dropout (SGD + dropout). We conduct an extensive grid search on the hyperparameter setting of these methods, and evaluate them on two popular continual learning benchmarks: permuted MNIST [175] and rotated MNIST datasets [179]. Permuted MNIST at each time step D_t consists of labeled MNIST images whose pixels undergo a fixed random permutation. Rotated MNIST is generated by the rotation of the original MNIST images. Figure 7.7a shows our method outperforms the other methods on the first four tasks, and is only slightly worse than

some (i.e., SGD for Permuted MNIST). Over the five tasks, our method achieves the highest average accuracies (95.16% on Rotated MNIST and 92.34% on permuted MNIST). Although some methods perform well on the fifth task after being trained on it, they fail to preserve the knowledge learned in previous tasks. In contrast, our method can preserve the knowledge from old tasks and achieves comparable or better accuracy on new tasks. Figure 7.7b shows the evolution of network structures for the five sequential tasks, as our method activates more neurons in the inferred hidden layers. In particular, the first hidden layer tends to be full from the first task with its activation probability close to one, the activation probabilities of the second and the third layers go up as the tasks are coming.

7.4.6 Case study on Genetic interaction inference

We apply our method on genetic interaction inference to show the improvements over heuristically designed neural network structure. The study of genetic interactions plays a crucial role in understanding biological phenomena and provides insight into the molecular etiology of diseases as well as the discovery of drug targets [8, 9].

We evaluate our method for biological network predictions as a binary classification task in terms of edges (genetic interactions) to be present or absent. Table 7.1 gives the number of genes and number of interactions for each dataset. The state-of-the-art method GNE [17] uses a single hidden layer with $M = 128$ neurons and a dropout rate of 0.5 to integrate interactions information with gene expression data for yeast and Ecoli dataset. We follow the model setting strategy similar to GNE and apply our model to infer the number of hidden layers and the number of activated neurons in each layer. We use $0.8 - 0.1 - 0.1$ train-validation-test split to split the interactions and randomly sample an equal number of negative interactions. Moreover, the prior is set to $\alpha = 1.1$ and $\beta = 3$ to encourage shallow model. We set the maximum number of neurons in each layer $M = 128$ and the truncation level $K^+ = 15$.

Table 7.1: Performance comparison of our method and GNE model with dropout regularization

	Yeast (5638, 977k)		**Ecoli** (3688, 294k)	
Method	**AUROC**	**AUPR**	**AUROC**	**AUPR**
GNE + dropout	0.825	0.821	0.940	0.939
Our method	**0.920 ± 0.005**	**0.933 ± 0.004**	**0.994 ± 0.001**	**0.994 ± 0.001**

Table 7.1 shows that the inferred network structure **Z** (depth and dropout regularization) outper-

forms the heuristically designed state-of-the-art method. For the Ecoli dataset, our model learns 3 hidden layers with 45, 15, and 4 active neurons, which improves AUROC by 11.5% and AUPR by 13.7%. Similarly, 2 hidden layers with 30 and 6 active neurons are inferred for yeast dataset that improves AUROC by 5.7% and AUPR by 5.86%. For both datasets, our model infers deeper and narrower neural network structure compared to GNE [17]. The result suggests that the appropriate choice of the number of hidden layers, and the number of neurons in each layer is crucial to design a neural network structure that achieves good performance. Our method employs Beta-Bernoulli processes to govern these parameters and infers the most plausible neural network structure warranted by data. This makes it applicable to tasks across various domains.

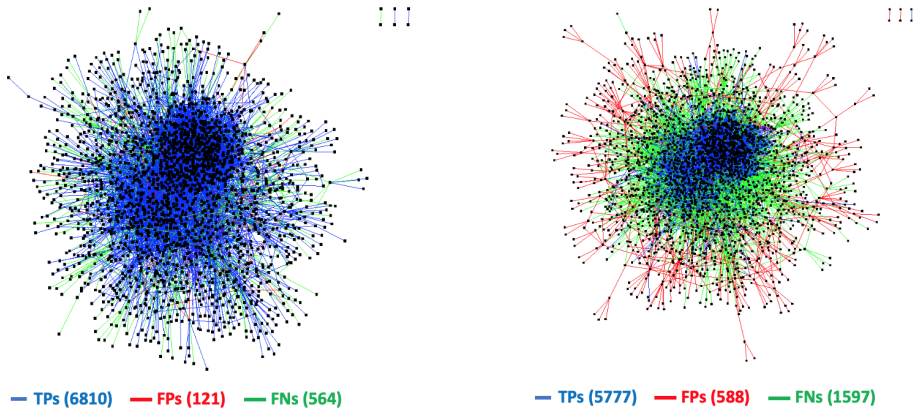


Figure 7.8: Genetic interaction network visualization for GNE with our method (left) and with dropout (right) for E-coli dataset. We use 0.5 as the threshold for the predicted probabilities. The black-colored nodes represent genes and the edges indicate their interactions. True positives (TPs), false negatives (FNs) and false positives (FPs) are highlighted and the counts are provided in the legend.

In Fig. 7.8, we plot the GI network inferred by GNE with our method and with dropout. Our method correctly identifies more true positives and predicts fewer false positives and false negatives than a dropout.

7.5 Conclusion

The proposed general joint inference framework can be applied to various neural networks. The experiments on MLPs and CNNs show that our method balances network depth and neuron activations to achieve superior performance. By enabling neural network structures to dynamically evolve to accommodate incrementally available data, we effectively alleviate catastrophic forgetting.

Chapter 8

Probabilistic Depth Selection for Graph Neural Network

Graph neural networks (GNNs) have achieved great success over recent years across various applications such as node classification, link prediction and graph classification. For link prediction, GNNs first compute node representations and aggregate representation of two nodes as the representation for edge/link. GNNs have been proposed to effectively learn representations for biomedical entities and achieved state-of-the-art results in biomedical interaction prediction. The success of GNNs in biomedical interaction prediction relies on the carefully designed and fine-tuned structure of GNNs, which is time-consuming and computationally expensive. Furthermore, deep GNNs with relatively large number of layers suffer from over-smoothing problem and thus indicate that the number of GNN layers is a critical parameter that have to be chosen carefully.

In Chapter 7, we developed a general framework for Bayesian Model Selection to jointly infer the most plausible depth warranted by data and dropout regularization simultaneously. We extend the framework to infer the depth for graph neural networks for biological interaction prediction.

8.1 Introduction

Biomedical interaction networks represent the complex interplay between heterogeneous molecular entities such as genes, proteins, drugs, and diseases within a biological system. The study of interaction networks advances our understanding of system-level understanding of biology [180]

and the discovery of biologically significant novel interactions including protein-protein interactions (PPIs) [120], drug-drug interactions (DDIs) [121], drug-target interactions (DTIs) [122] and gene-disease associations (GDIs) [123]. There have been huge technological advancements in high-throughput technologies that have produced ever-increasing amount of omics data and resulted in the identification of novel interactions. Despite this, current biological networks are noisy, sparse and incomplete, that limits our ability to study the biological phenomenon at system level.

Various computational methods have been developed to predict novel interactions in biomedical interaction networks. Recently, deep learning approaches on network datasets have shown great success across various domains such as social networks [124], recommendation systems [125], chemistry [126], citation networks [43]. Specifically, graph convolutional networks (GCNs) have shown great success in biomedical interaction prediction [19, 181]. GCN-based encoder aggregates the information from its immediate neighbors to learn the informative representation for each molecular entity and use these representations to predict biomedical interactions.

Despite the enormous success of GCN models in biomedical interaction prediction, the neural network structure of GCN models is a critical choice and needs to be carefully set. Most of the recent GCN models such as GCN [44] and GAT [182] define shallow network structures (*i.e.*, 2-layer models) to achieve their best performance. The ability of GCN models with shallow structures is limited and fails to extract information from higher-order neighbors. Although stacking multiple GCN layers and non-linearity enable the model to capture information from higher-order neighbors, such deep GCN models tend to face the over-smoothing problem as the performance of deep models degrades with the increase in the number of layers [44]. In particular, 2-layered GCN and GAT models outperform deep models.

To address the challenge, we propose Bayesian model selection [183] to jointly infer the depth of GCN models warranted by data and perform dropout regularization simultaneously [183]. To enable the number of GCN layers in the encoder to go to infinity in theory, we model the depth of GCN models as a stochastic process by defining the beta process over the number of GCN layers. The beta process induces layer-wise activation probabilities that modulate neurons activations for regularization via a conjugate Bernoulli process. The proposed framework balances the GCN model's depth and neuron activations and provides well-calibrated predictions.

We evaluate the performance of our inference framework and compare it with GCN-based models for biomedical interaction prediction. We demonstrate that the GCN-based encoder suffers from the over-fitting problem as the depth increases but our model infers the most plausible depth and is quite robust to the overfitting problem. Furthermore, experiments on sparse interaction networks with an

increasing number of interactions show that our method achieves superior performance by enabling the structure of the GCN-based encoder to dynamically evolve to accommodate incrementally available information.

In summary, our contributions are as follows:

- We propose a Bayesian model selection to jointly infer the most plausible depth and their neuron activations for encoder warranted by data simultaneously to learn representation for biomedical entities.
- We empirically demonstrate that our inference framework achieves superior performance by dynamically balancing the depth and their activation for the encoder.
- Our method enables the structure of the encoder to dynamically evolve to accommodate the information from an increasing number of interactions.
- We further show that our method is capable of making novel predictions that are validated using up-to-date literature-based database entries.

8.2 Materials and Methods

8.2.1 Biomedical interaction prediction

A biomedical network is a network with biomedical entities as nodes and their interactions as edges. Formally, a biomedical network can be defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ where $\mathcal{V}$ denotes the set of nodes representing biomedical entities such as proteins, genes, drugs, diseases and $\mathcal{E}$ represents the set of interaction between these entities. We follow the definition of biomedical interaction prediction in Definition 2.3.

8.2.2 Probabilistic model selection for graph convolutional networks

In this work, we propose a probabilistic model selection to infer the depth of graph convolutional encoder and their neurons activations to learn the representation for each entity in the network. In addition, we employ a bilinear decoder to reconstruct the edges in the interaction network. We adopt the encoder-decoder framework for biomedical interaction prediction [181]. Figure 8.1 shows the block diagram of the model.

- **Encoder**: a graph convolution encoder with potentially infinite hidden layers that takes an interaction graph $\mathcal{G}$ and generates representation for each entity in the interaction network. In particular, we propose to model the depth of graph convolution encoder as a stochastic process and jointly perform dropout regularization upon the inferred hidden layers.
- **Decoder**: a bilinear decoder that takes the representation of two nodes v_i and v_j and compute the probability of their interaction e_{ij}.

We next discuss the details of each component of the proposed framework.

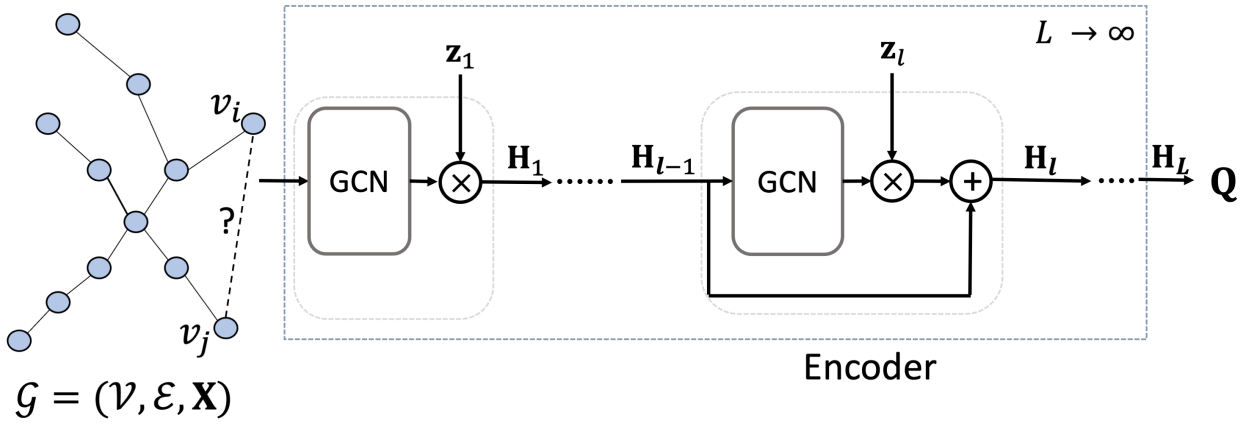


Figure 8.1: The block diagram of our prosed model with potentially infinite number of hidden layers in encoder. The input to the network is a biomedical interaction network $\mathcal{G}$ with edges $\mathcal{E}$ between entities $\mathcal{V}$ and feature matrix $\mathbf{X}$. The encoder with infinite hidden layers learns the final representation $\mathbf{Q}$. the bilinear decoder takes the representation $(\mathbf{q}_i, \mathbf{q}_j)$ for two entities (v_i, v_j) and predict the probability p_{ij} of their interaction based on the edge representation $\mathbf{e}_{ij}$.

Graph convolutional encoder with infinite layers

Graph convolution (GC) layer [44] generates representation for nodes in the network by repeatedly aggregating information from immediate neighbors. GC layer can be defined as:

$$\mathbf{H}_l = \sigma(\widehat{\mathbf{A}}\mathbf{H}_{l-1}\mathbf{W}_l) \tag{8.1}$$

where $\widehat{\mathbf{A}}$ is a symmetrically normalized adjacency matrix with self-connections $\widehat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_{|\mathcal{V}|})\mathbf{D}^{-\frac{1}{2}}$. Let $\mathbf{W}_l$ is a trainable weight matrix for layer l, $\mathbf{H}_{l-1}$ and $\mathbf{H}_l$ are the input and output

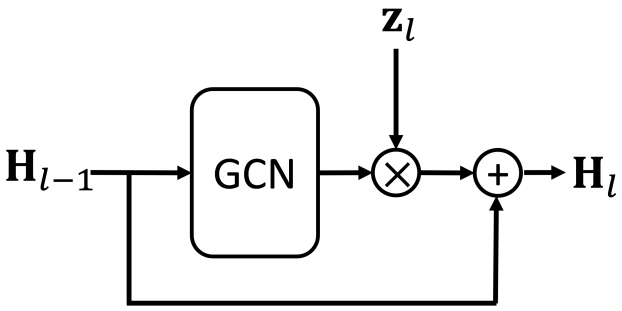


Figure 8.2: The block diagram for a layer in the encoder. The output $\mathbf{H}_{l-1}$ from the previous layer $l-1$ is passed into graph convolutional layer at layer l and the output from layer l is pruned by multiplying if with a binary vector $\mathbf{z}_l$ element wisely. Since there is a potentially infinite number of hidden layers in the encoder between the input and latent representation $\mathbf{Q}$, the skip connections pass the output from the last activated hidden layer to the bilinear decoder layer.

activations respectively. We can then define a GCN model with L layers can be defined as:

$$\mathbf{H}^{(l)} = \begin{cases} \mathbf{X} & \text{if } l = 0 \\ \sigma(\widehat{\mathbf{A}}\mathbf{H}_{l-1}\mathbf{W}_l) & \text{if } l \in [1, \ldots, L] \end{cases} \tag{8.2}$$

$\mathbf{H}_L \in \mathbb{R}^{|\mathcal{V}| \times D}$ represents the representation for each entity in the network. We denote this representation as $\mathbf{Q}$.

Deep GC encoder faces the issue of overfitting such that 2-layer models perform better than deeper models [44]. The depth (L) of the GC encoder is a critical choice and needs to be carefully set. To address this challenge, we propose to model the depth of encoder as a stochastic process by defining a beta process over the hidden layers [153, 156, 183]. We adopt the stick-breaking construction of the beta-Bernoulli process as:

$$z_{ml} \sim \text{Bernoulli}(\pi_l), \quad \pi_l = \prod_{j=1}^{l} \nu_j, \quad \nu_l \sim \text{Beta}(\alpha, \beta) \tag{8.3}$$

where ν_l are sequentially drawn from beta distribution and π_l represents the activation probability of layer l that decreases with layer l. z_{ml} represents Bernoulli variable with a probability that $z_{ml} = 1$ equals to π_l. Specifically, $z_{ml} = 1$ indicates that m'th neuron in layer l is activated. With

beta process over the hidden layers, the GC encoder takes the form:

$$\mathbf{H}_l = \sigma(\widehat{\mathbf{A}}\mathbf{H}_{l-1}\mathbf{W}_l) \bigotimes \mathbf{z}_l + \mathbf{H}_{l-1}, \quad l \in \{1, 2, \dots \infty\} \tag{8.4}$$

where $\mathbf{W}_l \in \mathbb{R}^{M \times m}$ is the weight matrix of layer l. We regularize the output of layer l by multiplying it elementwisely by a binary vector $\mathbf{z}_l$ where its element $z_{ml} \in \{0, 1\}$. Figure 8.2 shows the network structure with binary vector $\mathbf{z}_l$ applied to the output of layer l and skip connection.

Bilinear interaction decoder

We adopt bilinear decoder [181] to use representations obtained from the encoder and predict the probability of interactions between biomedical entities. Specifically, we define a bilinear layer that maps the representation of entities $\mathbf{q}_i \in \mathbb{R}^{M \times 1}$ and $\mathbf{q}_j \in \mathbb{R}^{M \times 1}$ to their edge representation $\mathbf{e}_{ij} \in \mathbb{R}^{d^* \times 1}$ as:

$$\mathbf{e}_{ij} = \text{ELU}(\mathbf{q}_i \mathbf{W}_b \mathbf{q}_j + \mathbf{b}) \tag{8.5}$$

where $\mathbf{e}_{ij}$ denotes the representation of edge between entities v_i and v_j, $\mathbf{W}_b \in \mathbb{R}^{d^* \times M \times M}$ represents the learnable fusion matrix, and $\mathbf{b}$ denotes the bias of bilinear layer. The probability p_{ij} of interaction between entities v_i and v_j is obtained by passing the edge representation $\mathbf{e}_{ij}$ through the fully-connected (FC) layer.

$$p_{ij} = \text{sigmoid}(\text{FC}_2(\text{ELU}(\text{FC}_1(\mathbf{e}_{ij})))) \tag{8.6}$$

8.2.3 Efficient variational approximation

Given an interaction dataset $D = \{\mathbf{A}, \mathbf{X}\}$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and feature matrix $\mathbf{X}$, we aim to reconstruct the edges in the input network. For the problem of classifying the edges, we express the likelihood of the neural network as:

$$p(D|\mathbf{Z}, \mathbf{W}) = \prod_{n=1}^{N} \text{Bernoulli}(A_{ij}|f(\mathbf{A}, \mathbf{X}; \mathbf{Z}, \mathbf{W})) \tag{8.7}$$

where $f(\cdot)$ represents the softmax function, $\mathbf{Z}$ is a binary matrix that represents the network structure for GC encoder whose l-th column is $\mathbf{z}_l$, and $\mathbf{W}$ denotes the set of weight matrices.

We define a prior over network structure $\mathbf{Z}$ via beta process as

$$\begin{aligned} p(\mathbf{Z}, \nu|\alpha, \beta) &= p(\nu|\alpha, \beta)p(\mathbf{Z}|\nu) \\ &= \prod_{l=1}^{\infty} \text{Beta}(\nu_l|\alpha, \beta)\text{Bernoulli}(z_{ml}|\pi_l) \end{aligned} \tag{8.8}$$

The marginal likelihood obtained by combining beta process prior in Equation 8.8 and the likelihood in Equation 8.7 is:

$$p(D|\mathbf{W}, L, \alpha, \beta) = \int p(D|\mathbf{W}, \mathbf{Z})p(\mathbf{Z}, \nu|\alpha, \beta) \tag{8.9}$$

The exact computation of marginal likelihood is intractable because of the non-linearity of the neural network and $L \to \infty$.

We then use a structured stochastic variational inference framework introduced by [154, 157] to approximate the marginal likelihood. The lower bound for log marginal likelihood is:

$$\begin{aligned} \log p(D|\mathbf{W}, L, \alpha, \beta) \geq \mathbb{E}_{q(\mathbf{Z},\nu)]}[\log p(D|\mathbf{W}, \mathbf{Z})] - \text{KL}[q(\nu)||p(\nu)] \\ - \text{KL}[q(\mathbf{Z}|\nu)||p(\mathbf{Z}|\nu)] \end{aligned} \tag{8.10}$$

where $q(\nu)$ and $q(\mathbf{Z}|\nu)$ represent variational beta distribution and variational Bernoulli distribution resepectively. Formally, we define $q(\nu) = \text{Beta}(\nu_k|a_k, b_k)$ with a_k and b_k as variational parameters. We next define $q(\mathbf{Z}|\nu)) = \text{ConcreteBernoulli}(z_{mk}|\pi_k)$. For variational approximation, we use truncation level K to denote potentially infinite number of layers.

Training algorithm

Our proposed framework leverages biomedical interaction network $\mathbf{A}$ and feature representation of biomedical entities $\mathbf{X}$. For all experiments, we set $\mathbf{X}$ to be one-hot encoding $\mathbf{I}_{|\mathcal{V}|}$. If features are available for biomedical entities, we can initialize $\mathbf{X}$ accordingly. We can also initialize the feature matrix using pre-trained embeddings from other network embedding approaches such as DeepWalk, node2vec.

To compute the expectation in Equation 8.10, we make Monte-Carlo estimation with S samples of encoder structure Z (encoder depth with layer-wise activation probabilities). The stick-breaking construction of beta-Bernoulli process induces that the probability of being activated in hidden layer l decreases exponentially with K. With a large K, a small number of hidden layers in the

encoder has activated neurons which can be obtained as

$$l^c = \max_l\{l| \sum_{m=1}^{M} z_{ml} > 1\} \tag{8.11}$$

where $\sum_{m=1}^{M} z_{ml}$ represents total activation of neurons in layer l. We can compute the expectation of log-likelihood based on S samples of encoder structure:

$$\mathbb{E}_{q(Z,v)}[\log p(D|\mathbf{Z}, \nu)] = \frac{1}{S}[\log p(D|\mathbf{Z}_s, \nu)] \tag{8.12}$$

We summarize the algorithm to train our proposed framework in Algorithm 2.

Algorithm 2 Training of our proposed method

Input $\{D_i\}_{i=1}^{B}$: B mini batches of interaction data

Input S: the number of samples of encoder structures $\mathbf{Z}$

1: **for** i $= 1, \ldots, B$ **do**
2: Draw S samples of encoder structures $\{\mathbf{Z}_s\}_{s=1}^{S}$ from $q(\mathbf{Z}, \boldsymbol{\nu})$
3: **for** s $= 1, \ldots, S$ **do**
4: Compute the number of layers l^c from $\mathbf{Z}_s$ using (8.11)
5: Compute $\log p(D_i|\mathbf{Z}_s, \mathbf{W})$ with l^c layers
6: Compute $\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}[\log p(D_i|\mathbf{Z}, \mathbf{W})]$ using (8.12)
7: Compute ELBO
8: Update $\{a_k, b_k\}_{k=1}^{l^c}$and $\{\mathbf{W}\}_{k=1}^{l^c}$ using backpropagation

The parameters of our proposed framework are learned by optimizing the ELBO in Equation 7.10 in an end-to-end manner. With the trained model, we can predict the probability of interaction between any pair of entities in the interaction network.

8.3 Results and discussion

We evaluate the performance of our proposed framework by applying it to infer the neural network structure for graph convolutional encoder on biomedical interaction prediction. We investigate our method's performance and compare it with the heuristically designed encoder structure. We further demonstrate that the settings of truncation

8.3.1 Biomedical interaction prediction

We evaluate various models on biomedical interaction prediction tasks using four interaction datasets such as protein-protein interactions (PPIs), drug-target interactions (DTIs), drug-drug interactions (DDIs), and gene-disease interactions (GDIs). In particular, we compare the performances of our proposed method with DeepWalk [40], node2vec [15], L3 [127], and graph convolutional network [44] with graph convolutional encoder and bilinear decoder.

For the prediction task, we split the interaction dataset in ratio 7:1:2 as a training, validation, and testing set. This procedure is repeated five times to generate five independent splits of the interaction dataset. We train all methods on the training dataset and evaluate their performance on the testing set. The validation dataset is used to select the best set of hyperparameters. The evaluation is done across five independent splits and the results with $\pm$ one standard deviation are reported in Table 8.1.

Table 8.1 shows that our proposed inference framework achieves significant improvement over other methods. In comparison to network embedding approaches such as DeepWalk and node2vec, our method gains 22.84% AUPRC gain in DTI, 40.83% in DDI, 26.85% in PPI, and 13.09% in GDI over DeepWalk. Although node2vec uses biased random walk and outperforms DeepWalk, our method achieves 19.97% AUPRC gain in DTI, 22.72% in DDI, 17.34% in PPI, and 12.82% in GDI over node2vec. We also observe that L3 outperforms all other methods but is limited to a specific aspect of network property, i.e., the path length of length 3 between two nodes in the network.

To investigate the advantage of inferring the most plausible network depth warranted by data, we compare our method with a predetermined GC encoder with 3 layers. We observe that the predetermined GC encoder achieves superior performance in comparison to network embedding methods such as DeepWalk, node2vec, and network similarity methods such as L3. We further observe that our proposed framework jointly infers the most plausible network structure for GC encoder and gains improvement over predetermined GC encoder. Specifically, our proposed framework achieves 3.35% AUPRC improvement in DTI, 2.29% in DDI, 1.45% in PPI, and 2.64% over GCN.

Since stacking more layers enable the encoder to capture information from high-order neighbors, the performance improvement indicates that our method aggregates information from the appropriate set of high-order neighbors by inferring the most plausible depth for graph convolutional encoder.

Table 8.1: Average AUPRC and AUROC with ± one standard deviation on biomedical interaction prediction

Dataset	Method	AUPRC	AUROC
DTI	DeepWalk	0.753 ± 0.008	0.735 ± 0.009
	node2vec	0.771 ± 0.005	0.720 ± 0.010
	L3	0.891 ± 0.004	0.793 ± 0.006
	GCN	0.896 ± 0.006	0.914 ± 0.005
	Ours	**0.925 ± 0.002**	**0.933 ± 0.002**
DDI	DeepWalk	0.698 ± 0.012	0.712 ± 0.009
	node2vec	0.801 ± 0.004	0.809 ± 0.002
	L3	0.860 ± 0.004	0.869 ± 0.003
	GCN	0.961 ± 0.005	0.962 ± 0.004
	Ours	**0.983 ± 0.002**	**0.982 ± 0.003**
PPI	DeepWalk	0.715 ± 0.008	0.706 ± 0.005
	node2vec	0.773 ± 0.010	0.766 ± 0.005
	L3	0.899 ± 0.003	0.861 ± 0.003
	GCN	0.894 ± 0.002	0.907 ± 0.006
	Ours	**0.907 ± 0.003**	**0.918 ± 0.002**
GDI	DeepWalk	0.827 ± 0.007	0.832 ± 0.003
	node2vec	0.828 ± 0.006	0.834 ± 0.003
	L3	0.899 ± 0.001	0.832 ± 0.001
	GCN	0.909 ± 0.002	0.906 ± 0.002
	Ours	**0.933 ± 0.001**	**0.945 ± 0.001**

8.3.2 Effect of truncation level K

We next investigate if the performance of the model with a predetermined graph convolutional encoder depends on the depth (L) of the encoder. Furthermore, we also evaluate the effect of the truncation level K on our model's performance.

For this experiment, we train predetermined encoder with different depths L over the range $L \in \{1, 5, 10, 15, 20\}$. Similarly, we train our proposed framework with truncation level K over the same range $K \in \{1, 5, 10, 15, 20\}$. For both methods, we set the maximum number of neurons D in each hidden layer in the encoder to 64. When $L = M = 1$, our method has an underfitting problem due to limited model capacity, although our method achieves better performance (Figure 8.3). When the truncation level K becomes sufficiently large ($K \geq 5$), the performance of our model does not

depend on K. Our method essentially works as the Bayesian model selection over the encoder depth which is consistent with the Theorem in [183].

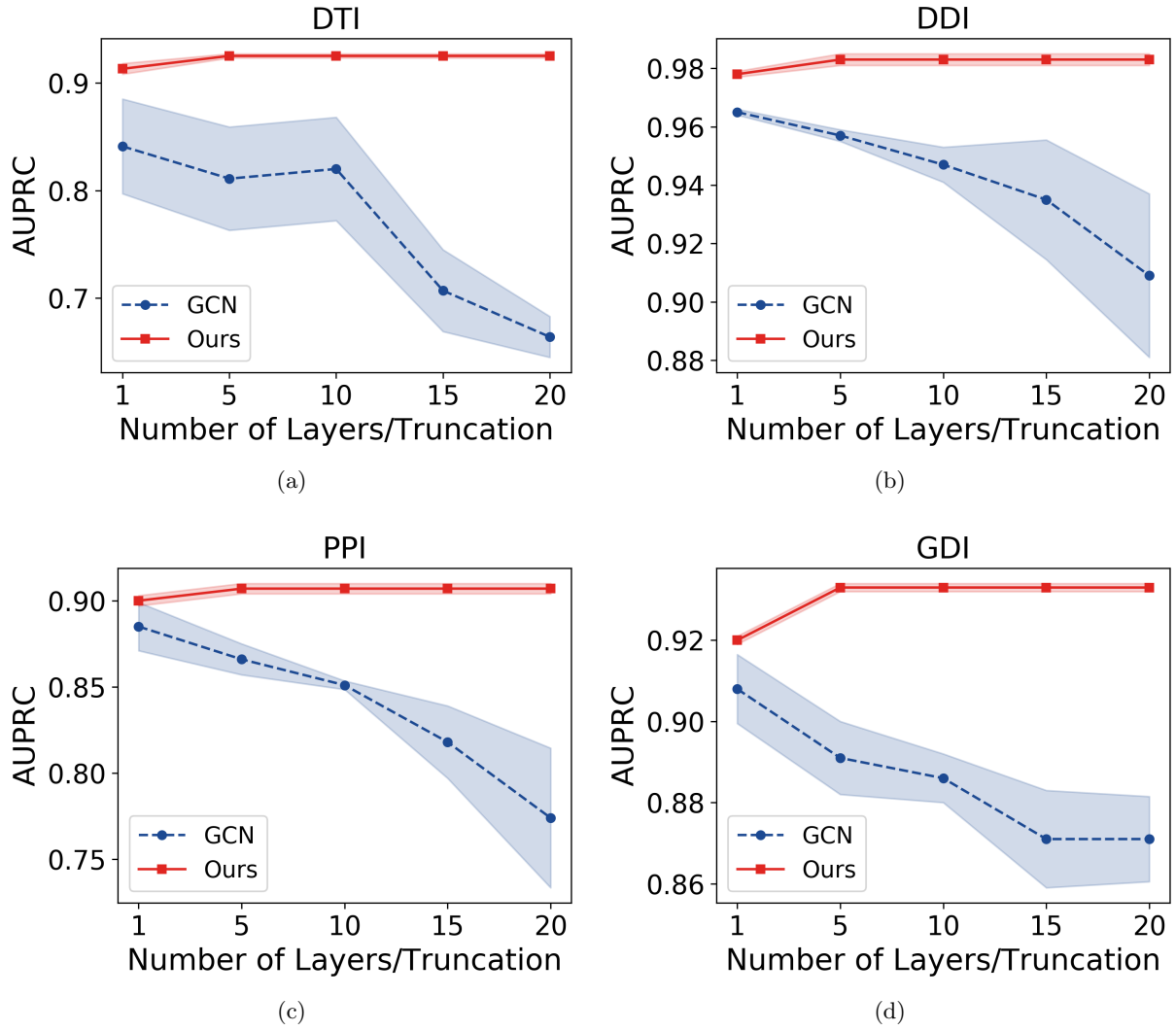


Figure 8.3: AUPRC comparison for our methods with different truncation and GCNs with different number of layers.

In contrast, the performance of the GC encoder depends on the depth to a great extent. Figure 8.3 shows that the GC encoder with predetermined depth suffers from an overfitting problem with the increase in the depth of the encoder. Shallow models with a single hidden layer perform better compared to deeper counterparts. The result in Figure 8.3 suggests that our method is quite robust to overfitting by jointly inferring the encoder depth and their neuron activations. In addition, our method balances the encoder depth and their network activations to achieve better performances.

Furthermore, Figure 8.4 shows the comparison of percentage of activated neurons with respect to truncation K or depth L. For GCN models with predetermined encoder structure, all neurons are activated and thus faces overfitting problem. On the contrary, our proposed method regularizes neuron activations and only activates relatively small number of neurons. With the settings $K = 20$, our model activates only 5% neurons.

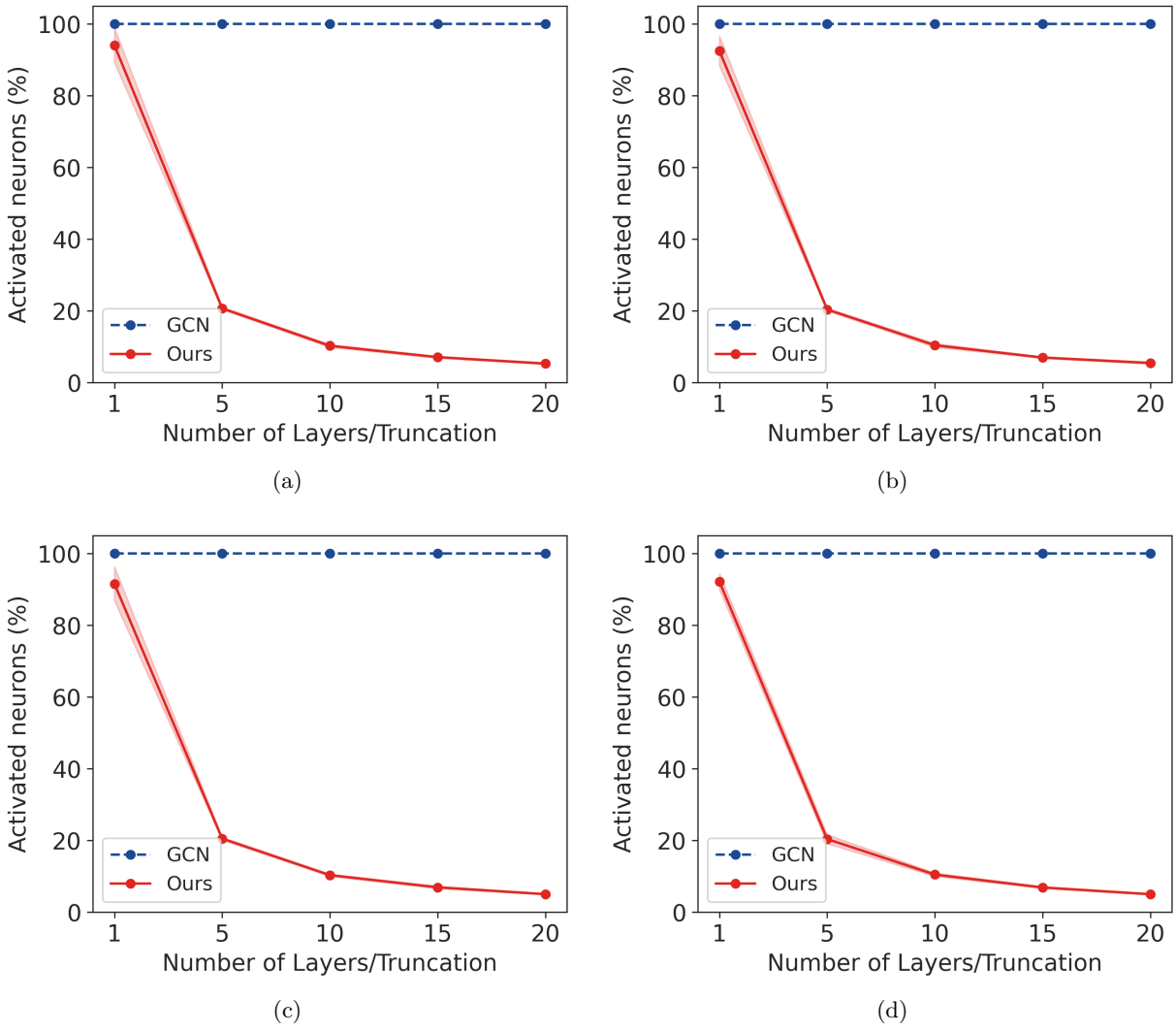


Figure 8.4: Comparison of activated neurons for our methods and GCNs when trained with different truncation K or depth L.

8.3.3 Calibrating model's prediction

We further evaluate if the predicted confidence p_{ij} represents the true likelihood of being true interaction. In this experiment, we expect the predicted confidence p_{ij} to be true interaction probability. To this aim, we compare our method with a predetermined graph convolutional network and compare the calibration results.

To compare the calibration of the model, we compute the Brier score [167] that is a proper scoring rule for measuring the accuracy of predicted probabilities. A lower Brier score represents better calibration of predicted probabilities. Mathematically, we compute Brier score as the mean squared error of the ground-truth interaction label A_{ij} and predicted probabilities p_{ij}:

$$\text{Brier score} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} (A_{ij} - p_{ij})^2 \tag{8.13}$$

where $|\mathcal{E}|$ is the number of edges in the test set.

Table 8.2: Brier scores for GCN and our method

Method	GCN	Ours
DTI	0.112 ± 0.001	$\mathbf{0.109 \pm 0.002}$
DDI	0.139 ± 0.003	$\mathbf{0.042 \pm 0.001}$
PPI	0.181 ± 0.046	$\mathbf{0.119 \pm 0.002}$
GDI	0.165 ± 0.035	$\mathbf{0.111 \pm 0.001}$

Table 8.2 shows the comparison of the Brier score obtained with the predetermined encoder and our inferred structure. Our method gains 2.68% in DTI, 69.78% in DDI, 34.25% in PPI, and 32.73% in GDIs. The results indicate that GCN with a predetermined encoder structure makes overconfident predictions. In contrast, our method achieves a lower Brier score, alluding to the benefits of inferring encoder structure for calibrated prediction. In conclusion, our method makes accurate and calibrated predictions for biomedical interaction.

8.3.4 Inferring encoder structure with respect to network sparsity

We next evaluate the robustness of our proposed method and GCN with a predetermined encoder. To this aim, we train the model on the varying percentage of training edges from 10% to 70%. We consider 20% of the interaction dataset as the test set to evaluate the predictions.

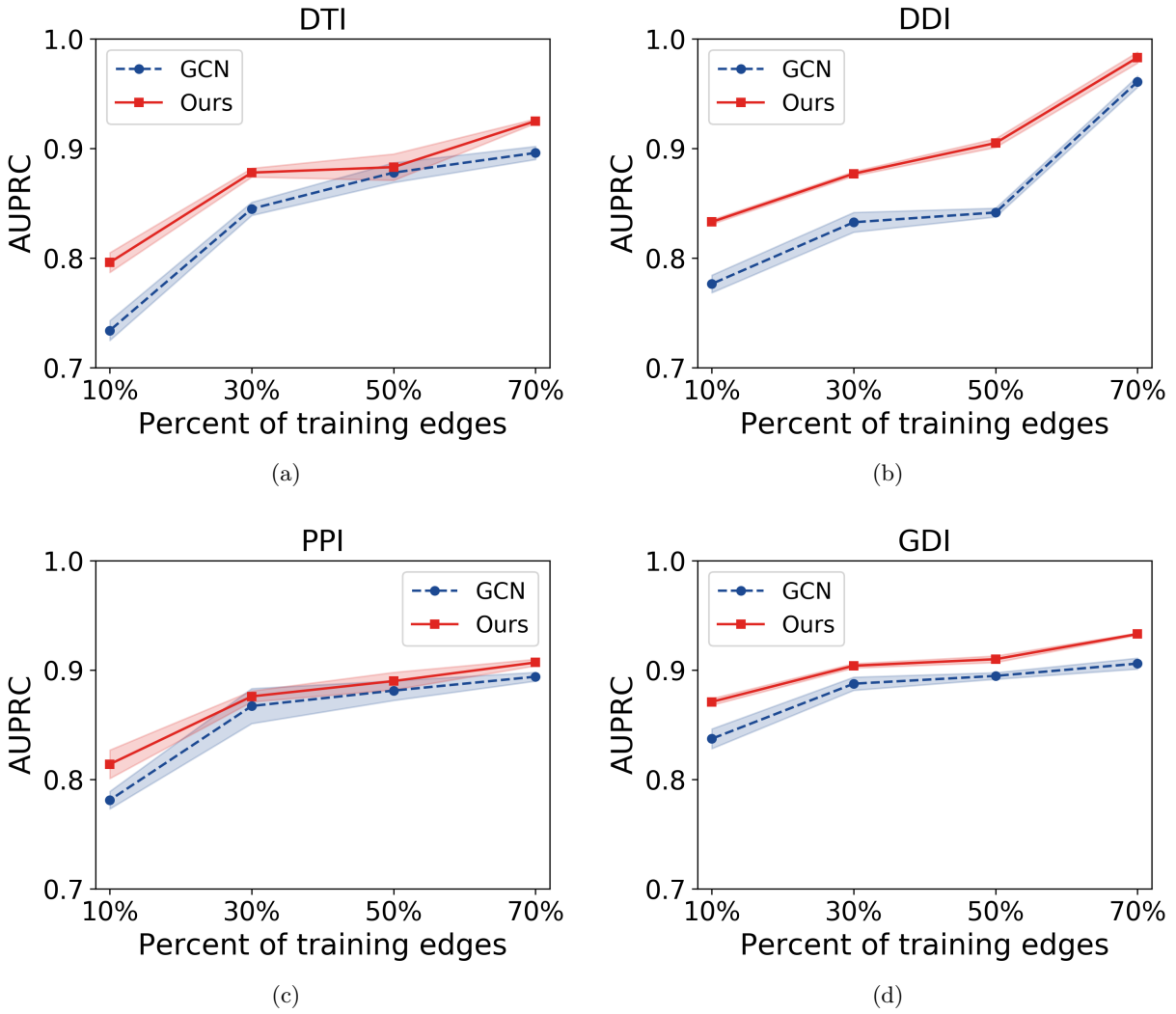


Figure 8.5: AUPRC comparison for our methods and GCNs when trained with different fractions of training edges.

Figure 8.5 shows that our proposed method is more robust compared to the alternative method. For all datasets, our model's performance increases with an increase in the number of training interactions. The inference of appropriate depth and their neuron activations enables our model to achieve better performance across all interaction dataset with varying sparsity.

In addition, our proposed method infers an appropriate structure for an encoder for biomedical interaction prediction. Figure 8.6 shows that the neural network structure dynamically evolve for DTI prediction task with increase in percentages of interactions, as our method activates more neurons in the inferred hidden layers. We observe similar behavior for all datasets.

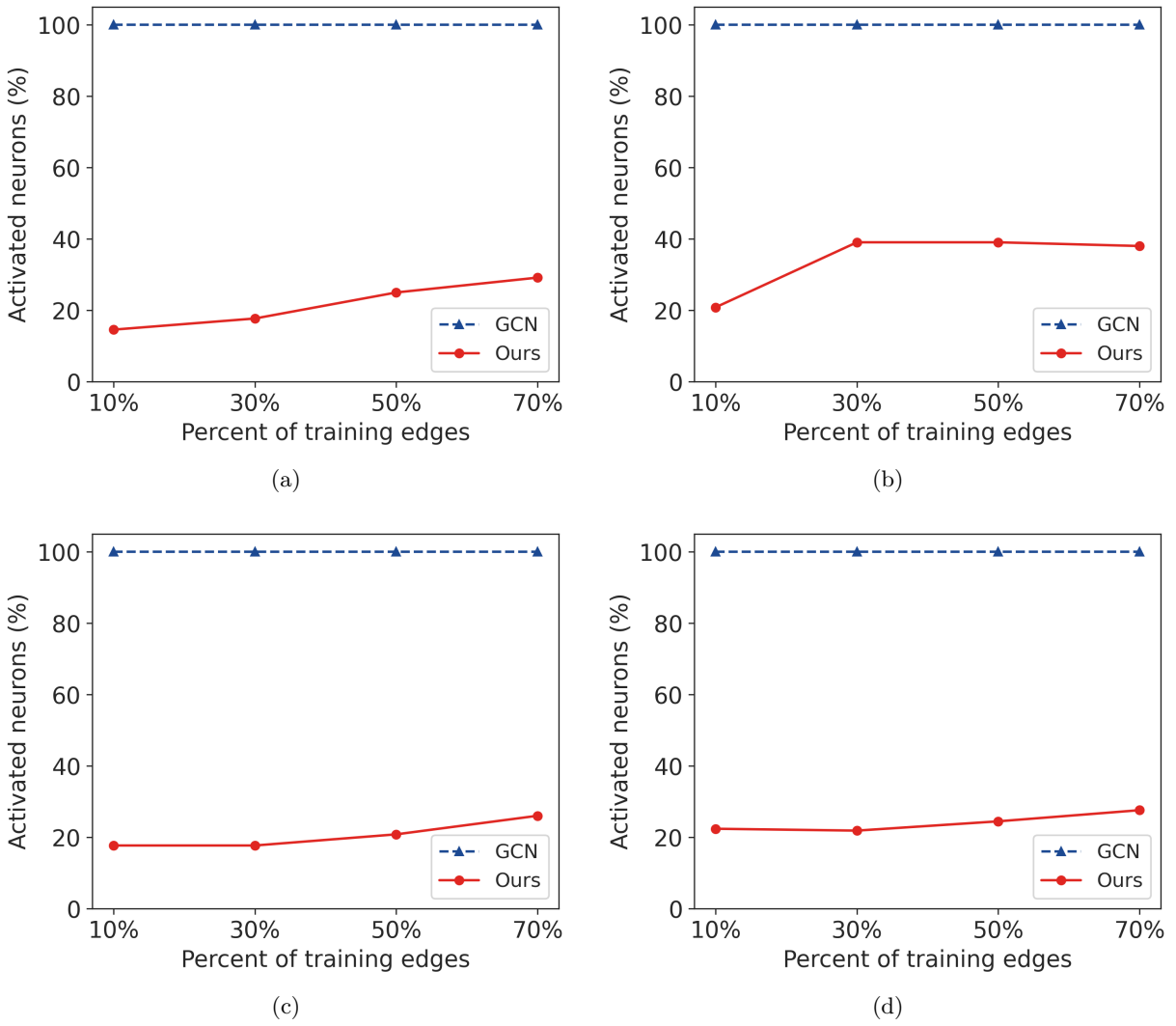


Figure 8.6: Comparison of activated neurons for our methods and GCNs when trained with different number of training interactions.

8.4 Conclusion

We present an inference framework to infer the structure of graph convolution-based encoder. The experiments on GC-based encoder for biomedical interaction prediction show that our method achieves accurate and calibrated predictions by balancing the depth of encoder and neuron activations. We further demonstrate that our framework enables the encoder's structure to dynamically evolve to accommodate the incrementally available number of interactions. Our inference framework infers a compact network structure of encoder with a relatively smaller number of parameters.

Chapter 9

Conclusion and Future works

Although high-throughput data generation and computational power have made tremendous progress over the past few decades, there are still many unresolved challenges to analyzing these ever-increasing heterogeneous data to identify potential links/interactions. Network-based computational approaches presented in this dissertation address the challenges such as integration of heterogeneous and high dimensional data, the inclusion of relational structure in the data, and the design of complex neural network structures appropriate for given data. To solve the problem in the biological domain, the relational inductive bias plays an important role by imposing constraints on relationships and interactions among entities in a learning process. By directly encoding the relational inductive biases present in the data, we develop the models that are data-efficient, scalable and also demonstrate the leap in their predictive power.

This dissertation focuses on the network-based computational approaches in application to biological network inference. The key contributions can be briefly summarized as:

- Design of deep neural network structures for interaction prediction using network representation of available interactions (Chapter 3), integration of continuous features with interaction network (Chaper 4) and integration of sequential features with network (Chapter 5).
- Design of higher-order graph convolutional network to capture local as well as higher-order network information from interaction network (Chapter 6).
- In Chapter 7, I have focused on the problem of inferring the neural network structure, realizing that the previously proposed methods require a predetermined neural network structure.

Experimental studies demonstrate the method's robustness, superior performance, and application to continual learning setup. Chapter 8 discusses the application of inference framework to infer the structure of the encoder for graph convolutional network.

9.1 Future works

Integration of heterogeneous omics data with higher-order graph convolutional network (HOGCN) In Chaper 6, the framework expects the node features $\mathbf{X}$ to be continuous. However, as discussed in Chapter 5, protein sequences are the most abundant data available for proteins. To include such features into the model, the sequences should be preprocessed to convert them to continuous features and information may be lost during this phase. The goal is to extend HOGCN to take various types of data such as sequences or images to enable end-to-end learning.

Inference of network neighborhood for higher-order graph convolutional network The order of HOGCN in Chapter 6 is a hyperparameter and needs to be carefully set. The hyperparameter search needs to be performed to identify the best order but this process is time-consuming and expensive for large datasets. The next step is to develop a novel probabilistic model selection approach to infer the appropriate order for HOGCN.

Interpretability of graph-based deep learning methods for interaction prediction Interpretability of deep neural networks is crucial for their practical applications in the biological domain. However, none of the current methods can provide such interpretability for biological interaction prediction. Besides the good performance of the model, it is also crucial for graph-based neural networks to be interpretable and explainable that help domain experts understand the inner working of the model. A future direction is to enable the model to generate explanations for the prediction to answer the questions such as "what contributes to the prediction" and "why the model makes certain predictions".

Biological network with heterogeneous nodes and edges types A broader future direction is to focus on biological networks of heterogeneous entities with different types of edges/interactions (Figure 9.1). The representation of biological systems as heterogeneous networks opens the door for a lot of potential research avenues. The heterogeneous network provides a more complete representation of a biological network compared to a network with a single type of nodes or edges.

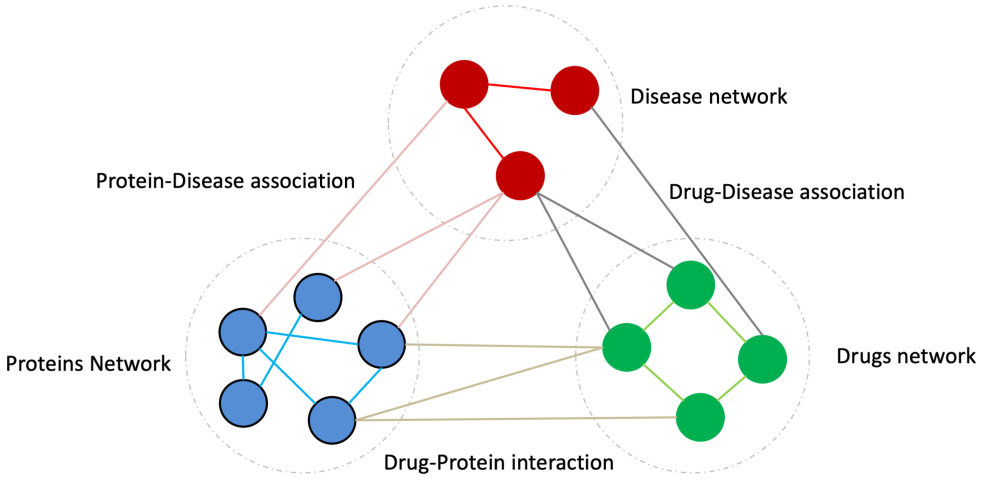


Figure 9.1: A network representation with different biological entities and different interaction types.

Bibliography

[1] Juho Lee, Saehoon Kim, Jaehong Yoon, Hae Beom Lee, Eunho Yang, and Sung Ju Hwang. Adaptive network sparsification with dependent variational beta-bernoulli dropout. *arXiv preprint arXiv:1805.10896*, 2019.

[2] Jiashi Feng and Trevor Darrell. Learning the structure of deep convolutional networks. In *Proc. of the IEEE International Conference on Computer Visions (ICCV)*, page 2749–2757, 2015.

[3] Suraj Srinivas and Venkatesh Babu. Learning neural network architectures using backpropagation. In *Proc. of the British Machine Vision Conference (BMVC)*, pages 104.1–104.11, 2016.

[4] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *Proc. of the 14th European Conference on Computer Vision (ICCV)*, pages 646–661, 2016.

[5] Georgi Dikov and Justin Bayer. Bayesian learning of neural network architectures. In *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 730–738, 2019.

[6] Eric Nalisnick, Jose Miguel Hernandez-Lobato, and Padhraic Smyth. Dropout as a structured shrinkage prior. In *Proc. of the 36th International Conference on Machine Learning (ICML)*, pages 4712–4722, 2019.

[7] Javier Antorán, James Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[8] Ramamurthy Mani, Robert P St Onge, John L Hartman, Guri Giaever, and Frederick P Roth. Defining genetic interaction. *Proceedings of the National Academy of Sciences*, 105(9):3461–3466, 2008.

[9] Benjamin Boucher and Sarah Jenna. Genetic interaction networks: better understand to better predict. *Frontiers in genetics*, 4:290, 2013.

[10] Kasper Lage. Protein–protein interactions and genetic diseases: the interactome. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1842(10):1971–1980, 2014.

[11] Shawn Martin, Diana Roe, and Jean-Loup Faulon. Predicting protein–protein interactions using signature products. *Bioinformatics*, 21(2):218–226, 2004.

[12] Hans Peter Fischer. Mathematical modeling of complex biological systems: from parts lists to understanding systems behavior. *Alcohol Research & Health*, 31(1):49, 2008.

[13] Natalia B Janson. Non-linear dynamics of biological systems. *Contemporary Physics*, 53(2):137–168, 2012.

[14] Yu Li, Chao Huang, Lizhong Ding, Zhongxiao Li, Yijie Pan, and Xin Gao. Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. *Methods*, 166:4–21, 2019.

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

[16] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[17] KC Kishan, Rui Li, Feng Cui, Qi Yu, and Anne R Haake. Gne: a deep learning framework for gene network inference by aggregating biological information. *BMC systems biology*, 13(2):38, 2019.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[19] Kexin Huang, Cao Xiao, Lucas Glass, Marinka Zitnik, and Jimeng Sun. Skipgnn: Predicting molecular interactions with skip-graph networks. *arXiv preprint arXiv:2004.14949*, 2020.

[20] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, 50:71–91, 2019.

[21] Marinka Žitnik and Blaž Zupan. Data imputation in epistatic maps by network-guided matrix completion. *Journal of Computational Biology*, 22(6):595–608, 2015.

[22] Jörg Menche, Amitabh Sharma, Maksim Kitsak, Susan Dina Ghiassian, Marc Vidal, Joseph Loscalzo, and Albert-László Barabási. Uncovering disease-disease relationships through the incomplete interactome. *Science*, 347(6224), 2015.

[23] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

[24] Xiang Yue, Zhen Wang, Jingong Huang, Srinivasan Parthasarathy, Soheil Moosavinasab, Yungui Huang, Simon M. Lin, Wen Zhang, Ping Zhang, and Huan Sun. Graph embedding on biomedical networks: Methods, applications and evaluations. *Bioinformatics*, 36(4):1241–1251, 2020.

[25] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[26] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions, 2020.

[27] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, et al. Pfam: the protein families database. *Nucleic acids research*, 42(D1):D222–D230, 2014.

[28] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29, 2019.

[29] Abhijeet R Sonawane, Scott T Weiss, Kimberly Glass, and Amitabh Sharma. Network medicine in the age of biomedical big data. *Frontiers in Genetics*, 10, 2019.

[30] Eric de Silva and Michael PH Stumpf. Complex networks and simple models in biology. *Journal of the Royal Society Interface*, 2(5):419–430, 2005.

[31] Michael E Cusick, Niels Klitgord, Marc Vidal, and David E Hill. Interactome: gateway into systems biology. *Human molecular genetics*, 14(suppl_2):R171–R181, 2005.

[32] Nikolas Papanikolaou, Georgios A Pavlopoulos, Theodosios Theodosiou, and Ioannis Iliopoulos. Protein–protein interaction predictions using text mining methods. *Methods*, 74:47–53, 2015.

[33] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[34] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[35] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[36] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[37] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.

[38] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[40] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[41] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.

[42] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, volume 16, pages 1145–1152, 2016.

[43] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[44] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[45] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

[46] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[48] Suraj Srinivas and R Venkatesh Babu. Generalized dropout. *arXiv preprint arXiv:1611.06791*, 2016.

[49] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proc. of the 30th International Conference on Machine Learning (ICML)*, pages 1058–1066, 2013.

[50] Song Han, Jeff Pool, Jeff Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, page 1135–1143, 2015.

[51] John Ingraham and Debora Marks. Variational inference for sparse and undirected models. In *Proc. of the 34th International Conference on Machine Learning (ICML)*, pages 1607–1616, 2017.

[52] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of the 33rd International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

[53] Durk P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, page 2575–2583, 2015.

[54] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proc. of the 34th International Conference on Machine Learning (ICML)*, pages 2498–2507, 2017.

[55] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 3581–3590, 2017.

[56] Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, and Eric P. Xing. Nonparametric variational auto-encoders for hierarchical representation learning. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 5094–5102, 2017.

[57] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[59] Konstantinos P. Panousis, Sotirios Chatzis, and Sergios Theodoridis. Nonparametric bayesian deep networks with local competition. In *Proc. of the 36th International Conference on Machine Learning (ICML)*, pages 4980–4988, 2019.

[60] Hyunghoon Cho, Bonnie Berger, and Jian Peng. Compact integration of multi-network topology for functional analysis of genes. *Cell systems*, 3(6):540–548, 2016.

[61] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[62] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[63] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of International Conference on Learning Representations*, 2015.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[65] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[66] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539, 2006.

[67] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[68] Ying-Ke Lei, Zhu-Hong You, Zhen Ji, Lin Zhu, and De-Shuang Huang. Assessing and predicting protein interactions by combining manifold embedding with multiple information integration. In *BMC bioinformatics*, volume 13, page S3. BioMed Central, 2012.

[69] Yadi Zhou, Yuan Hou, Jiayu Shen, Yin Huang, William Martin, and Feixiong Cheng. Network-based drug repurposing for novel coronavirus 2019-ncov/sars-cov-2. *Cell discovery*, 6(1):1–18, 2020.

[70] Deisy Morselli Gysi, Ítalo Do Valle, Marinka Zitnik, Asher Ameli, Xiao Gan, Onur Varol, Helia Sanchez, Rebecca Marlene Baron, Dina Ghiassian, Joseph Loscalzo, et al. Network medicine framework for identifying drug repurposing opportunities for covid-19. *arXiv preprint arXiv:2004.07229*, 2020.

[71] Allan Peter Davis, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, Roy McMorran, Jolene Wiegers, Thomas C Wiegers, and Carolyn J Mattingly. The comparative toxicogenomics database: update 2019. *Nucleic acids research*, 47(D1):D948–D954, 2019.

[72] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[73] David S Wishart, Craig Knox, An Chi Guo, Savita Shrivastava, Murtaza Hassanali, Paul Stothard, Zhan Chang, and Jennifer Woolsey. Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic acids research*, 34(suppl_1):D668–D672, 2006.

[74] Damian Szklarczyk, Andrea Franceschini, Stefan Wyder, Kristoffer Forslund, Davide Heller, Jaime Huerta-Cepas, Milan Simonovic, Alexander Roth, Alberto Santos, Kalliopi P Tsafou, et al. String v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic acids research*, 43(D1):D447–D452, 2015.

[75] Neel S Madhukar, Olivier Elemento, and Gaurav Pandey. Prediction of genetic interactions using machine learning and network properties. *Frontiers in bioengineering and biotechnology*, 3:172, 2015.

[76] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *arXiv preprint arXiv:1705.04969*, 2017.

[77] Stephen Oliver. Proteomics: guilt-by-association goes global. *Nature*, 403(6770):601, 2000.

[78] Y Tu, G Stolovitzky, and U Klein. Quantitative noise analysis for gene expression microarray experiments. *Proceedings of the National Academy of Sciences*, 99(22):14031–14036, 2002.

[79] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402. ACM, 2005.

[80] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[81] Daniel Marbach, James C Costello, Robert Küffner, Nicole M Vega, Robert J Prill, Diogo M Camacho, Kyle R Allison, Andrej Aderhold, Richard Bonneau, Yukun Chen, et al. Wisdom of crowds for robust gene network inference. *Nature methods*, 9(8):796, 2012.

[82] Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Biocomputing 2000*, pages 418–429. World Scientific, 1999.

[83] Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong. Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics*, 22(13):1623–1630, 2006.

[84] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[85] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[86] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[87] Fenglou Mao, Phuongan Dam, Jacky Chou, Victor Olman, and Ying Xu. Door: a database for prokaryotic operons. *Nucleic acids research*, 37(suppl_1):D459–D463, 2008.

[88] Jason E Miller, Liye Zhang, Haoyang Jiang, Yunfei Li, B Franklin Pugh, and Joseph C Reese. Genome-wide mapping of decay factor–mrna interactions in yeast identifies nutrient-responsive transcripts as targets of the deadenylase ccr4. *G3: Genes, Genomes, Genetics*, 8(1):315–330, 2018.

[89] Jason Liang, Namit Singh, Christopher R Carlson, Claudio P Albuquerque, Kevin D Corbett, and Huilin Zhou. Recruitment of a sumo isopeptidase to rdna stabilizes silencing complexes by opposing sumo targeted ubiquitin ligase activity. *Genes & development*, 31(8):802–815, 2017.

[90] Mohan Babu, Cedoljub Bundalovic-Torma, Charles Calmettes, Sadhna Phanse, Qingzhou Zhang, Yue Jiang, Zoran Minic, Sunyoung Kim, Jitender Mehla, Alla Gagarinova, et al. Global landscape of cell envelope protein complexes in escherichia coli. *Nature biotechnology*, 36(1):103, 2018.

[91] Michael C Gustin, Jacobus Albertyn, Matthew Alexander, and Kenneth Davenport. Map kinase pathways in the yeastsaccharomyces cerevisiae. *Microbiology and Molecular biology reviews*, 62(4):1264–1300, 1998.

[92] Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[93] Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences*, 104(11):4337–4341, 2007.

[94] Lei Yang, Jun-Feng Xia, and Jie Gui. Prediction of protein-protein interactions from protein sequence using local descriptors. *Protein and Peptide Letters*, 17(9):1085–1090, 2010.

[95] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic acids research*, 36(9):3025–3030, 2008.

[96] Zhu-Hong You, Lin Zhu, Chun-Hou Zheng, Hong-Jie Yu, Su-Ping Deng, and Zhen Ji. Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set. In *BMC bioinformatics*, volume 15, page S9. BioMed Central, 2014.

[97] Sam Tonddast-Navaei and Jeffrey Skolnick. Are protein-protein interfaces special regions on a protein's surface? *The Journal of chemical physics*, 143(24):12B631_1, 2015.

[98] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. Predicting protein-protein interactions through sequence-based deep learning. *Bioinformatics*, 34(17):i802–i810, 09 2018.

[99] Muhao Chen, Chelsea Ju, Guangyu Zhou, Xuelu Chen, Tianran Zhang, Kai-Wei Chang, Carlo Zaniolo, and Wei Wang. Multifaceted protein-protein interaction prediction based on siamese residual rcnn. *Bioinformatics*, 35(14):i305–i314, 07 2019.

[100] Tobias Hamp and Burkhard Rost. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics*, 31(12):1945–1950, 2015.

[101] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[102] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. of International Conference on Machine Learning*, pages 1614–1623, 2016.

[103] Vlad Niculae and Mathieu Blondel. A regularized framework for sparse and structured neural attention. In *Proc. of Advances in Neural Information Processing Systems*, pages 3338–3348, 2017.

[104] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *Proc. of International Conference on Learning Representations*, 2018.

[105] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. Deep variational network embedding in wasserstein space. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2827–2836. ACM, 2018.

[106] Wenting Liu and Jagath C Rajapakse. Fusing gene expressions and transitive protein-protein interactions for inference of gene regulatory networks. *BMC systems biology*, 13(2):37, 2019.

[107] Clark R Givens, Rae Michael Shortt, et al. A class of wasserstein metrics for probability distributions. *The Michigan Mathematical Journal*, 31(2):231–240, 1984.

[108] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. *Predicting Structured Data*, 1:0, 2006.

[109] Christophe Dessimoz, Toni Gabaldón, David S Roos, Erik LL Sonnhammer, Javier Herrero, and Quest for Orthologs Consortium. Toward community standards in the quest for orthologs. *Bioinformatics*, 28(6):900–904, 2012.

[110] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2018.

[111] Tobias Hamp and Burkhard Rost. More challenges for machine-learning protein interactions. *Bioinformatics*, 31(10):1521–1525, 2015.

[112] Rose Oughtred, Chris Stark, Bobby-Joe Breitkreutz, Jennifer Rust, Lorrie Boucher, Christie Chang, Nadine Kolas, Lara O'Donnell, Genie Leung, Rochelle McAdam, et al. The biogrid interaction database: 2019 update. *Nucleic acids research*, 47(D1):D529–D541, 2018.

[113] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[114] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[115] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[116] Qian Cong, Ivan Anishchenko, Sergey Ovchinnikov, and David Baker. Protein interaction networks revealed by proteome coevolution. *Science*, 365(6449):185–189, 2019.

[117] Minoru Kanehisa. Linking databases and organisms: Genomenet resources in japan. *Trends in biochemical sciences*, 22(11):442–444, 1997.

[118] Sonja Baumli, Sabine Hoeppner, and Patrick Cramer. A conserved mediator hinge revealed in the structure of the med7· med21 (med7· srb7) heterodimer. *Journal of Biological Chemistry*, 280(18):18171–18178, 2005.

[119] Eric F Pettersen, Thomas D Goddard, Conrad C Huang, Gregory S Couch, Daniel M Greenblatt, Elaine C Meng, and Thomas E Ferrin. Ucsf chimera—a visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612, 2004.

[120] Katja Luck, Dae-Kyum Kim, Luke Lambourne, Kerstin Spirohn, Bridget E Begg, Wenting Bian, Ruth Brignall, Tiziana Cafarelli, Francisco J Campos-Laborie, Benoit Charloteaux, et al. A reference map of the human binary protein interactome. *Nature*, 580(7803):402–408, 2020.

[121] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[122] Yunan Luo, Xinbin Zhao, Jingtian Zhou, Jinglin Yang, Yanqing Zhang, Wenhua Kuang, Jian Peng, Ligong Chen, and Jianyang Zeng. A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nature communications*, 8(1):1–13, 2017.

[123] Monica Agrawal, Marinka Zitnik, Jure Leskovec, et al. Large-scale analysis of disease pathways in the human interactome. In *PSB*, pages 111–122. World Scientific, 2018.

[124] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 2110–2119, 2018.

[125] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 974–983, 2018.

[126] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272, 2017.

[127] István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. *Nature communications*, 10(1):1–8, 2019.

[128] Iftikhar Ahmad, Muhammad Usman Akhtar, Salma Noor, and Ambreen Shahnaz. Missing link prediction using common neighbor and centrality based parameterized algorithm. *Scientific Reports*, 10(1):1–9, 2020.

[129] Ozlem Keskin, Nurcan Tuncbag, and Attila Gursoy. Predicting protein–protein interactions from the molecular to the proteome level. *Chemical reviews*, 116(8):4884–4909, 2016.

[130] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

[131] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. High-order proximity preserved embedding for dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 30(11):2134–2144, 2018.

[132] Ryan A. Rossi, Nesreen K. Ahmed, and Eunyee Koh. Higher-order network representation learning. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 3–4, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[133] Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi Yadkori. Hone: higher-order network embeddings. *arXiv preprint arXiv:1801.09303*, 2018.

[134] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[135] Marinka Zitnik, SM Rok Sosic, and Jure Leskovec. Biosnap datasets: Stanford biomedical network dataset collection. *http://snap.stanford.edu/biodata*, 2018.

[136] Janet Piñero, Juan Manuel Ramírez-Anguita, Josep Saüch-Pitarch, Francesco Ronzano, Emilio Centeno, Ferran Sanz, and Laura I Furlong. The disgenet knowledge platform for disease genomics: 2019 update. *Nucleic acids research*, 48(D1):D845–D855, 2020.

[137] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative

style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[138] Diederik P Kingma and Jimmy Ba. Adam (2014), a method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR), arXiv preprint arXiv*, volume 1412, 2015.

[139] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[140] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[141] B Garcia, P De Juana, T Bermejo, J Gómez, P Rondon, I Garcia Plaza, et al. Sequential interaction of ritonavir and nelfinavir with acenocoumarol (abstract 1069). In *Seventh European Conference on Clinical Aspects and Treatment of HIV Infection. Lisbon, Portugal*, 1999.

[142] Emilio Perucca. Clinically relevant drug interactions with antiepileptic drugs. *British journal of clinical pharmacology*, 61(3):246–255, 2006.

[143] Elizabeth Landrum Michalets and Charlene Rhinehart Williams. Drug interactions with cisapride. *Clinical pharmacokinetics*, 39(1):49–75, 2000.

[144] Teresa Truong and James Haley. Probable warfarin and dapsone interaction. *Clinical and Applied Thrombosis/Hemostasis*, 18(1):107–109, 2012.

[145] Berna Ozbek and Gulten Otuk. Post-antibiotic effect of levofloxacin and tobramycin alone or in combination with cefepime against pseudomonas aeruginosa. *Chemotherapy*, 55(6):446–450, 2009.

[146] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, pages 9244–9255, 2019.

[147] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. volume 33, 2020.

[148] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proc. of the Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

[149] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, page 3290–3300, 2017.

[150] Sida Wang and Christopher Manning. Fast dropout training. In *Proc. of the 30th International Conference on Machine Learning (ICML)*, pages 118–126, 2013.

[151] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

[152] Romain Thibaux and Michael I Jordan. Hierarchical beta processes and the indian buffet process. In *Proc. of the Artificial Intelligence and Statistics (AISTATS)*, pages 564–571, 2007.

[153] Tamara Broderick, Michael I. Jordan, and Jim Pitman. Beta processes, stick-breaking and power laws. *Bayesian Analysis*, 7(2):439–476, Jun 2012.

[154] Matthew Hoffman and David Blei. Stochastic structured variational inference. In *Proc. of the Artificial Intelligence and Statistics (AISTATS)*, pages 361–369, 2015.

[155] Rachit Singh, Jeffrey Ling, and Finale Doshi-Velez. Structured variational autoencoders for the beta-bernoulli process. In *NIPS Workshop on Advances in Approximate Bayesian Inference*. 2017.

[156] John Paisley, Aimee Zaas, Christopher W. Woods, Geoffrey S. Ginsburg, and Lawrence Carin. A stick-breaking construction of the beta process. In *Proc. of the 27th International Conference on Machine Learning*, pages 2902–2911. JMLR. org, 2010.

[157] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[158] Kai Xu, Akash Srivastava, and Charles Sutton. Variational russian roulette for deep bayesian nonparametrics. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 6963–6972, 2019.

[159] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

[160] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

[161] Hagai Attias. A variational bayesian framework for graphical models. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, page 209–215, 1999.

[162] Martin Jankowiak and Fritz Obermeyer. Pathwise derivatives beyond the reparameterization trick. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 2235–2244, 2018.

[163] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *Proc. of the 35th International Conference on Machine Learning (ICML)*, page 1436–1445, 2018.

[164] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[165] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proc. of the International Conference on Machine Learning*, pages 1861–1869. PMLR, 2015.

[166] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Proc. of the Advances in Neural Information Processing Systems*, 30, 2017.

[167] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[168] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[169] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[170] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[171] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[172] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Proc. of the Advances in Neural Information Processing Systems*, volume 32, 2019.

[173] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2015.

[174] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[175] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proc. of the national academy of sciences*, 114(13):3521–3526, 2017.

[176] Seyed Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. Dropout as an implicit gating mechanism for continual learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 232–233, 2020.

[177] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[178] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3762–3773. PMLR, 2020.

[179] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proc. of the 24th International Conference on Machine learning (ICML)*, pages 473–480, 2007.

[180] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18(9):551, 2017.

[181] Kishan KC, Rui Li, Feng Cui, and Anne Haake. Predicting biomedical interactions with higher-order graph convolutional networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2021.

[182] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[183] Kishan KC, Rui Li, and Mahdi Gilany. Joint inference for neural network depth and dropout regularization. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.