

Rapid #: -23666276

CROSS REF ID: **1028904**

LENDER: **NTE (Washington State University) :: Main Library**

BORROWER: **EYW (Wayne State University) :: Main Library**

TYPE: Article CC:CCG

JOURNAL TITLE: Applied soft computing

USER JOURNAL TITLE: Applied Soft Computing

ARTICLE TITLE: Reconstruction of gene regulatory networks using graph neural networks

ARTICLE AUTHOR: M., Emma

VOLUME: 163

ISSUE:

MONTH:

YEAR: 2024

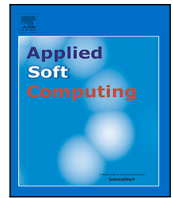
PAGES: 111899-

ISSN: 1568-4946

OCLC #:

Processed by RapidX: 12/10/2024 3:00:07 PM

This material may be protected by copyright law (Title 17 U.S. Code)



Reconstruction of gene regulatory networks using graph neural networks

Emma Paul M. *, Jereesh A.S., G. Santhosh Kumar

Bioinformatics Lab, Department of Computer Science, Cochin University of Science and Technology, Kalamassery, 682022, India

ARTICLE INFO

Keywords:

Gene expression
Gene regulatory network
Graph neural network
Semi-supervised edge classification

ABSTRACT

Gene regulatory network (GRN) inference, a longstanding challenge in computational biology, aims to construct GRNs from genomic data. Graph Neural Networks (GNNs) are well-suited for this task due to their ability to leverage both node features and topological relationships. This research systematically evaluated various GNN variants, gradually narrowing the focus through a filtering process. The study considered multiple design aspects, including layers, epochs, decoders, activation functions, graph structures, aggregation methods, skip connections, dropout, and hidden dimensions. Ultimately, two promising models emerged, one based on the Chebyshev spectral graph convolutional operator and the other on the Hypergraph convolutional operator, demonstrating state-of-the-art performance. Notably, hypergraphs demonstrated superior performance on real datasets with higher-order dependencies, while the Chebyshev model showed greater generalization across both simulated and real datasets. The code for this research is available online at <https://github.com/EmmaDPaul/GRN-inference-using-GNN>.

GITHUB site

1. Introduction

Gene Regulatory Network (GRN) inference is a major research area in computational biology. Genes code for a large number of products that our body requires such as proteins, and RNA (Ribonucleic Acid); the products of one gene can influence another one and can even involve self-loops affecting the very same gene from which it all started [1]. Inferring this entire regulatory network interaction from gene expression data using experimentation is infeasible. Computational methods strategically reduce the search space to the most probable sets of relations through gene expression data analysis like correlation-based analysis [2], network inference methods like Bayesian Networks [3] and Differential Equation Models [4], which can then be experimentally verified. This enables speeding up the advancement of disease prognosis prediction.

In this work, we approach the problem of GRN inference by treating it as a link prediction task within Graph Neural Networks (GNNs). The primary motivation for employing GNNs lies in their ability to utilize the features of two genes and their respective neighbors to predict whether a link exists between them. The novelty of the study is that it infers GRNs using a wide range of GNN convolution layer variants on multiple benchmark datasets. Our contributions in this paper encompass the following aspects:

1. Development of a semi-supervised edge classification framework for GRN inference.

2. Conducted a comprehensive comparison of GNN variants, activation functions, decoder functions, and feature augmentation techniques, providing insights into their impact on GRN inference accuracy.
3. The framework was rigorously tested on a variety of simulated and real datasets, to assess performance and generalization capabilities.

The paper includes an introduction to the context, providing a brief overview of established techniques for constructing GRNs. It then delves into the materials and methods, which encompass the dataset and the methodology employed. Subsequently, the results and discussion section offers a comparison between the proposed approach and current methods. Finally, the paper concludes by mentioning the limitations of the study, and a summary.

2. Background

There is a large number of unsupervised methods for GRN inference. Many methods have been proposed over the years that come under statistical and machine learning-based techniques. The summarized information on background techniques along with examples are provided in Fig. 1.

* Corresponding author.

E-mail addresses: emmapaul1993@cusat.ac.in (Emma Paul M.), jereesh@cusat.ac.in (Jereesh A.S.), san@cusat.ac.in (G. Santhosh Kumar).

<https://doi.org/10.1016/j.asoc.2024.111899>

Received 25 September 2023; Received in revised form 20 May 2024; Accepted 18 June 2024

Available online 25 June 2024

1568-4946/© 2024 Elsevier B.V. All rights reserved, including those for text and data mining, AI training, and similar technologies.

Category	Methods	Remarks	Disadvantages
Information theoretic approaches	MRNET (Meyer et al., 2007)	Applies MRMR principle for feature selection, good performance in benchmark datasets	MRNET is susceptible to noise, may oversimplify regulatory relationships within GRNs,
Regression Models	GENIE3 (Huynh-Thu et al., 2010), GENIE3-time Huynh-Thu (2012), KBOOST (Iglesias-Martinez et al., 2021), BiXGBoost (Zheng et al., 2019), BTNET (Park et al., 2018), JUMP3 (Huynh-Thu and Sanguinetti, 2015), TIGRESS (Haury et al., 2012)	Considers one target gene at a time and recognizing potential regulators that best predict the target gene state.	Can be less interpretable, may not consider natural network structure.
Support vector machines (SVMs)	SIRENE (Mordelet and Vert, 2008), CompareSVM (Gillani et al., 2014), GRADIS (Razaghi-Moghadam and Nikoloski, 2020)	Utilize SVMs for learning interaction classes	SIRENE: Local knowledge approach less effective, CompareSVM: Limited to small networks, GRADIS: Requires prior knowledge of TFs
Causality based techniques	GreyNet (Chen and Liu, 2022), SWING (Finkle et al., 2018), GranReg (Paul M. et al., 2023), Patil and Vaida (2022)	Identifies causal relationships	Relies on stationary time series data (unrealistic for GRNs)
Link prediction heuristics	Common neighbors (Liben-Nowell and Kleinberg, 2003), Katz index (Lü and Zhou, 2011), Adamic-Adar index (Zhou et al., 2009), PageRank (Page, 1997)	Simple to implement	Success depends on the specific network
Other techniques			
Ensemble techniques	Wisdom of Crowds (Marbach et al., 2012)	Improved performance and robustness to noise	Increased complexity, limited interpretability, base model dependency
Bayesian network models	BGRMI (Iglesias-Martinez et al., 2016), FBISC (Narimani et al., 2017), KBOOST (Iglesias-Martinez et al., 2021), IMBDANET (Liu et al., 2022), Richards et al., 2021.	Integrates variables and their conditional interdependencies	Computationally expensive for large networks, relies on assumptions of probabilistic independence
Deep learning approaches	Mandal et al., 2015, MacLean, 2019, Shrivastava, 2023	Handling complexity, scalability	Lack of interpretability

Fig. 1. Popular GRN inference techniques.

2.1. Information theoretic approaches

Our study explores information theoretic approaches for inferring GRNs, with a focus on the MRNET [5] method. MRNET stands out for its application of the maximum relevance/minimum redundancy (MRMR) principle in feature selection within a supervised learning framework. By selecting features that minimize redundancy with existing ones while maximizing mutual information with the target variable, MRNET identifies gene relations using microarray data. However, like other methods in this category, MRNET is susceptible to noise and may oversimplify intricate regulatory relationships within GRNs, potentially overlooking interaction directionality.

2.2. Regression models

Regression models utilize regression and feature selection methods to deal with network inference by considering **one target gene at a time and recognizing potential regulators that best predict the target gene state** [6]. The main works under this category are provided in Fig. 1. GENIE3 employs random forests to discern the genes influencing the target gene. As the victor of the DREAM4 challenge, GENIE3 serves as a widely used benchmark technique in the field. GENIE3-time is an extension tailored for time series data, where the expression of a gene at time point $t + 1$ is regarded as dependent on the gene expressions at time point t . One of the negatives of random forests is that there is very

little understanding of how the feature selection with random forest is performed as the internal working of individual trees and ensemble can be difficult to interpret. TIGRESS employs a randomization technique similar to GENIE3 for scoring gene interactions; however, they differ in their aggregation methods. GENIE3 utilizes feature aggregation from decision trees, while TIGRESS employs the least angle regression technique. In terms of results, TIGRESS initially provides superior predictions with a set threshold but exhibits increasing errors further down the list of interactions. The JUMP3 technique originates from the idea of creating a hybrid approach, combining the advantages of both model-free and model-based techniques. Starting with an on/off biological model, it reconstructs edges using a tree-based approach inspired by GENIE3. The jump tree decision function concept is grounded in a probabilistic model. However, JUMP3 may encounter challenges in precisely capturing time-lagged dependencies, particularly in time series data, and it can be computationally expensive. BiXGBoost identifies regulatory and target genes for a specific gene using a bidirectional BiXGBoost model. On the other hand, BTNET takes time-series data as input to a boosted tree and computes a weighted adjacency matrix to determine regulatory interactions between gene pairs. Both techniques are based on gradient-boosting algorithms. It is important to note that boosting techniques, including these, are less interpretable, given their ensemble of trees, and they do not inherently consider the natural network structure of a GRN.

2.3. Support vector machines (SVMs)

SIRENE employs SVMs to learn the classes of target and nontarget genes associated with a Transcription Factor (TF). It utilizes genes with no reported interaction with the TF as a negative set, which is then divided for training and testing purposes. The use of local knowledge, where a classifier is constructed for each TF to distinguish targets from nontargets, is deemed less effective compared to a global approach. The global approach considers all TF-target pairs for classifier creation, making it applicable to any other TF-gene pair. CompareSVM is another technique, that leverages SVMs to assess various SVM kernel methods on simulated datasets. Its evaluation has been conducted on networks with a node size below 200, and as the node size increases (around 500), the choice of inference relies on the nature of the experimental condition. GRADIS is a supervised method that utilizes a network-based portrayal of gene expression data. This more detailed representation is utilized to differentiate between TF-gene pairs that interact and those that do not, surpassing the mere capture of the relationship between a TF and a gene. It is important to note, however, that GRADIS necessitates prior knowledge of TFs for optimal utilization and entails the construction of features.

2.4. Causality based techniques

While numerous machine learning approaches have been suggested, a majority of them operate as black boxes, lacking interpretability. GreyNet [7] introduces a sliding window technique based on grey theory to identify gene regulatory interactions. These associations are subsequently transformed into causal relationships using Granger causality regression techniques, establishing directional regulatory links. However, this process presents challenges, given the potential existence of cyclic feedback loops and multiple causal structures. SWING [8] employs a windowed model based on multivariate Granger causality to assess several regulators using time series datasets across various time point delays. A seasonality differencing preprocessing followed by regression-Granger causality-based was introduced in GranReg [9]. The approach introduced by Patil and Vaida [10] employs a deep learning framework based on Granger causality for inferring GRNs. This framework leverages graph convolutions and Long Short-Term Memory to uncover causal relationships from gene expression data. However, a significant drawback of employing Granger causality is its reliance on stationary time series datasets, which proves to be a challenging assumption to meet in the context of GRNs.

2.5. Other techniques

Wisdom of Crowds uses ensemble learning and combines six unsupervised approaches for prediction tasks [11]. The major benefits of using ensemble techniques are improved performance and robustness to noise. However, it also comes with disadvantages such as increased complexity, limited interpretability, and base model dependency. Bayesian network models incorporate variables and their conditional interdependencies through Directed Acyclic Graphs. A significant limitation of Bayesian network models is their computational complexity, particularly evident when handling large networks featuring numerous variables. Additionally, these models rely on assumptions of probabilistic independence among variables, and any deviation from these assumptions may result in biased outcomes. Deep learning approaches have also emerged for the inference of GRNs. Mandal [12] proposes a neural network-based model that uses neural networks to reconstruct small-scale GRN from gene expression data. Another technique by MacLean [13] utilizes a convolutional neural network on pairs of Arabidopsis TF and their respective targets, with microarray gene expression serving as the feature set. Shrivastava's [14] study centers on accentuating the disparity between models for sparse graph recovery and their practical applications in the inference of GRNs. The research outlines potential avenues for reconstructing GRNs using sparse graph recovery models. The key focus of the study revolves around categorizing graph recovery models into four main types; regression, Markov network, Graphical Lasso, and Directed Acyclic Graph-based models. In neural network-based approaches, significant drawbacks include complexity and a lack of interpretability.

2.6. Link prediction heuristics

Heuristics that use score functions to predict links come under this class, examples include the number of common neighbors, Katz index [15,16]. The score generated by this heuristic relies on the number of hops between a node and its neighboring nodes, resulting in various variants. The simplest heuristics begin with a one-hop neighborhood, exemplified by common neighbors and preferential attachment [17]. The Adamic-Adar index considers two hops [18], while some other methods, such as Katz and PageRank [19], utilize the entire network for scoring [20]. One of the earliest endeavors in the automatic learning of predefined graph structural features was presented by Zhang and Chen [21]. The success of the heuristic usually depends on the particular network it is being used on [22].

2.7. GNN models

The majority of supervised GRN methods concentrate on pairwise gene analysis, often losing sight of the broader network context. Some research has highlighted the significance of local subgraphs in decoding valuable information related to the existence of links [23,24]. **A major benefit of the GNN is its capability to model complex relationships not only based on gene pairs but also based on graph neighborhoods.** In GNNs, it is possible to have both node attributes and edge characteristics. Leveraging these attributes can enhance the performance of the GNN, and the extent of improvement depends on the specific dataset as factors like complexity, noise, and clear patterns can influence the final result. Unlike tabular and image data, graph data has interesting properties which are quite difficult to work with. Graph structure is non-Euclidean and they lay differently in space. The normal metrics for Euclidean space will not fare well in such datasets. The GNN considers node attributes, edge attributes, and the global arrangement of the graph to generate embeddings that characterize the graph's structure [25]. These node embeddings contain information encompassing structural aspects and features derived from neighboring nodes. The building block of the GNNs is the message-passing layers which combine the node, edge, and adjacency information and convert

it to node embeddings. Graph convolution is the technique used in message-passing layers and has different variations [26]. Using message passing a node aggregates neighbor node information and combines that with its information. This is performed for every node to create the node embeddings [27]. The number of layers in the GNN defines the total count of message passing. Each message-passing layer can be considered as a hop to the neighbors, the total number of message-passing layers thus denotes the number of hops done from a node to its neighborhood [27]. When it comes to score functions mentioned in link prediction heuristics some perform better on certain networks than others. So rather than learning predefined heuristics, it is often preferred to learn the heuristic from the network directly. After considering the target link, the adjacent local subgraph is leveraged for predicting the link's presence, contributing to an enhanced understanding of link prediction mechanisms. In this way, the heuristics are determined on the go. SEAL proposes a solution where small hops are used to learn higher-order features utilizing the GNN [23]. GRGNN [24] is an approach for constructing GRNs using GNNs. It is an extension of SEAL and is applicable in both supervised and semi-supervised frameworks. In this approach, GRN inference is framed as a graph classification task. Positive and negative subgraphs are generated based on the presence or absence of links with TFs. The node features are derived from both gene expression and graph information. GRINCD [28], Q-GAT [29] are some other works in this area related to GRN inference. **GNN applied to graph data involves three main types of predictions; node prediction, link prediction, and graph prediction.** Link prediction applications encompass recommender systems, knowledge graph completion [30], and network reconstruction. **GRN inference falls into the category of link prediction.**

3. Materials and methods

The GRN which is reconstructed from gene expression data can be represented using $G = (V, E)$ where V represents the vertices and E represents the edges. G is a directed graph where edge $e_{ij} \in E$, represents the interaction between v_i to v_j which in turn represents the regulatory link from gene g_i to gene g_j and $E \subseteq V \times V$.

3.1. Dataset

Microarray [31] keeps track of thousands of genes in parallel and produces output raw files in image format which can then be transformed into gene expression matrices after data pre-processing. **In these matrices the table rows represent samples and the column represents the genes; the number in the cell characterizes the value of that specific gene in the corresponding row sample.** While static data has no time component, temporal also known as time-series data have interactions that vary across time. Additionally, there are other types of data, such as knockout and knockdown. Knockout involves the removal of a specific gene to gather insights into its function, whereas the knockdown entails reducing the expression of a specific gene. In this study, the input comprises datasets from the Dialogue for Reverse Engineering Assessments and Methods (DREAM), encompassing DREAM3, DREAM4 [32–34] and DREAM5 [11] benchmark datasets. In this study, we operate under the assumption that GRNs adhere to a scale-free topology, as suggested by Ouma et al. [35]. The details of the datasets are provided in Table 1. In this research, we focus on analyzing graph structures that are both directed and homogeneous.

3.2. Preprocessing

The datasets of DREAM3 and DREAM4 challenges with a size of 10 were excluded from consideration due to their limited network size. The study commences with the utilization of the DREAM3 size 50 dataset. Since GNNs necessitate a graph-based input, the initial step involved distinguishing between node features and edge features to

construct the graph structure. As a result, four distinct graph variations were created using the DREAM3 size 50 dataset. Utilizing graphs as input allows for greater flexibility, as data elements can be represented as nodes and edges with associated attributes. This approach enables the incorporation of diverse information, including static values, time-dependent sequences, and augmented values. Table 2 provides detailed information about these four preprocessed graphs.

For DREAM3 and DREAM4 datasets time series data with 21 time points is provided from which all four graph structures were made. For DREAM5 datasets time series data is not available hence for DREAM5 only two graph structures were created.

Knockdown and Knockout data were used as edge features by finding the difference between wild-type values and knockdown or knockout values of a gene with one another.

To analyze the effect of adding predefined node features, four augmented features were added namely degree, betweenness centrality, clustering coefficient, and page rank. Degree denotes the number of edges a node is connected to, betweenness centrality shows the influence a node has on the information flow of the graph, and clustering coefficient is the ratio of actual links to the maximum link possible in a neighborhood for a node. Page rank calculates the importance of a node based on incoming links from other nodes. The influence of the source node of a link also plays a role in determining a node's Page Rank score.

3.3. Pipeline

For transductive link prediction, auto-encoders have achieved great success [26,36]. Here we have utilized an encoder–decoder architecture for the prediction of GRN links. The major components of the pipeline are as follows.

By using multi-layer convolutional networks, the encoder creates node embeddings by processing the input graph. For the DREAM3 size 50 dataset, 23 convolutional layers were tested. Out of which the top four performers were used on DREAM4 size 100 genes and DREAM5 datasets. The convolution layers were chosen after careful consideration of the challenges often encountered in gene regulatory network inference. These challenges include discovering complex relationships, such as interactions between multiple genes, and the rapid increase in the size and types (static, temporal) of gene expression data. Additionally, the need to integrate diverse datasets, encompassing genomic, transcriptomic, proteomic, and metabolomic data, poses a significant hurdle [37,38]. The node embedding was done using, 2, 3, 4, and 5-layer combinations. Eight different activation functions were tried out along with five different aggregation methods. For the implementation of GNNs, we have utilized PyG [39].

The decoder calculates score predictions on edges using node embeddings, the predictions are made for both positive and negative edges. The positive edges denote interactions between genes and the negative edges denote no interaction. Three types of decoders were used to obtain the link scores. They are dot product, neural network, and cos decoder. Negative links are added to make it to binary classification. The architecture used is provided in Fig. 2. In each layer, there is both message passing from nearby nodes and its aggregation which can be generally represented as Eq. (1) [41].

$$agg = AGGREGATE^{(k)}(\{h_v^{(k)}, \forall v \in N(u)\}) \quad (1)$$

For a node u , the $AGGREGATE$ function takes in the states of all direct neighbors v and aggregates them in a specific way. Here $N(u)$ represents the set of neighboring nodes of u . The layer index is denoted k , with $k \in \{1, \dots, K\}$ where K denotes the total number of layers in the GNN. The $UPDATE$ function then combines the aggregated neighborhood information to the current state of node u , as shown in Eq. (2) [41]. There are different variants of GNNs derived by changing the $AGGREGATE$ and $UPDATE$ functions.

$$h_u^{(k+1)} = UPDATE^{(k)}(h_u^{(k)}, agg) \quad (2)$$

Table 1

Datasets used for the study, with the number of genes represented by node count, existing interactions represented by edge count, Type of dataset: simulated/real, and Organism. Among the datasets DREAM3 size 50, DREAM4 size 100, and InSilico, E. coli, and S. cerevisiae datasets from DREAM5 were used for the analysis.

Dataset name	Network	Node count	Edge count	Type	Organism
DREAM3	Network 1	50	62	Sim	E. coli, Yeast
	Network 2		82		
	Network 3		77		
	Network 4		160		
	Network 5		173		
	Network 1	100	125	Sim	E. coli, Yeast
	Network 2		119		
	Network 3		166		
	Network 4		389		
	Network 5		551		
DREAM4	Network 1	100	176	Sim	E. coli and S. cerevisiae
	Network 2		249		
	Network 3		195		
	Network 4		211		
	Network 5		193		
DREAM5	Network 1	1643	4012	Sim	E. coli, Yeast
	Network 2	2810	518	Real	S.aureus
	Network 3	4511	2066	Real	E. coli
	Network 4	5950	3940	Real	S. cerevisiae

Table 2

The table delineates four input graphs derived from datasets, offering details on the content, designated names, and specifics of node and edge attribute inclusion for each.

Input graph	Name used	Information included
Basic graph	basic_data_	Node features: node id, wild-type values. Edge features: Knockout, Knockdown values
Basic graph + Time series gene expression	basic_TS_data_	Node features: node id, wild-type values, Time series Edge features: Knockout, Knockdown values
Basic graph + Augmented node features	basic_aug_data_	Node features: node id, wild-type values, Augmented node features Edge features: Knockout, Knockdown values
Basic graph + Time series gene expression + Augmented node features	basic_TS_aug_data_	Node features: node id, wild-type values, Time series, Augmented node features Edge features: Knockout, Knockdown values

The computational module of the general design pipeline for the GNN module consists of 3 core modules namely the propagation, sampling, and pooling [40].

3.3.1. Propagation operators

The propagation operator can be further divided into convolution operator, and skip connection. There are two approaches namely spectral and spatial under the convolution operator. In spectral approaches, graphs are converted to spectral representations. In spectral methods, spectral techniques are based on graph signal processing [42]. The Graph Fourier Transformation of signal s is done before the convolution operation, the resulting signal is then transformed back using inverse Graph Fourier Transform (Eq. (3)). In this case, the gene expression values present in the nodes act as signals. For $G = (V, E, W)$ where V denotes vertices, E is the set of edges and $W \in R^{n \times n}$ is the adjacency matrix of the graph. Here n represents the number of nodes in the graph and each node in turn corresponds to a gene [40].

$$\mathcal{F}(s) = U^T s \quad (3a)$$

$$\mathcal{F}^{-1}(s) = U s \quad (3b)$$

In Eq. (3), U is the matrix of eigenvectors of normalized graph Laplacian L , $U \in R^{n \times n}$. Normalized graph Laplacian L is given by Eq. (4) [40].

$$L = I_n - D^{-1/2} W D^{-1/2} \quad (4)$$

where I_n is the identity matrix and D is the degree matrix. The Laplacian L is diagonalized by Fourier basis U as represented by Eq. (5) [40].

$$L = U \Lambda U^T \quad (5)$$

where $\Lambda = \text{diag}([\lambda_0, \lambda_1, \dots, \lambda_{n-1}]) \in R^{n \times n}$. λ represents real nonnegative eigenvalues identified as frequencies of the graph. On the basis of convolution theorem [43], the convolution operation is defined as Eq. (6).

$$y = g_\theta(L)x = g_\theta(U \Lambda U^T) x = U g_\theta(\Lambda) U^T x \quad (6)$$

where x denotes the input signal, y denotes the output signal and g_θ denotes the filter function applied in the Fourier domain. There are different spectral methods based on the filter variant it uses. ChebConv uses polynomial filters, as defined in Eq. (7) [44].

$$g_\theta(\Lambda) = \sum_{y=0}^{Y-1} \theta_y \Lambda^y \quad (7)$$

where θ is a vector of polynomial coefficients of Y th order polynomial. Following this recursive computation of $g_\theta(L)$ represented as Chebyshev polynomial function is done to reduce the cost of filtering signal s . In the graph coarsening stage multilevel clustering algorithms are used to produce coarser graphs to view the data domain in a different resolution. After coarsening the vertices are arranged using a balanced binary tree. ChebNet [45], GCN [26], and AGCN [46] are some of the other methods coming under this category.

Basic spatial approaches such as GraphSAGE [47] use graph topology to define the convolutions. The major challenge of such approaches is the varying neighborhood size and maintenance of local invariance property. GraphSAGE samples and aggregates features to generate embeddings. There are two main parts, one is the embedding generation

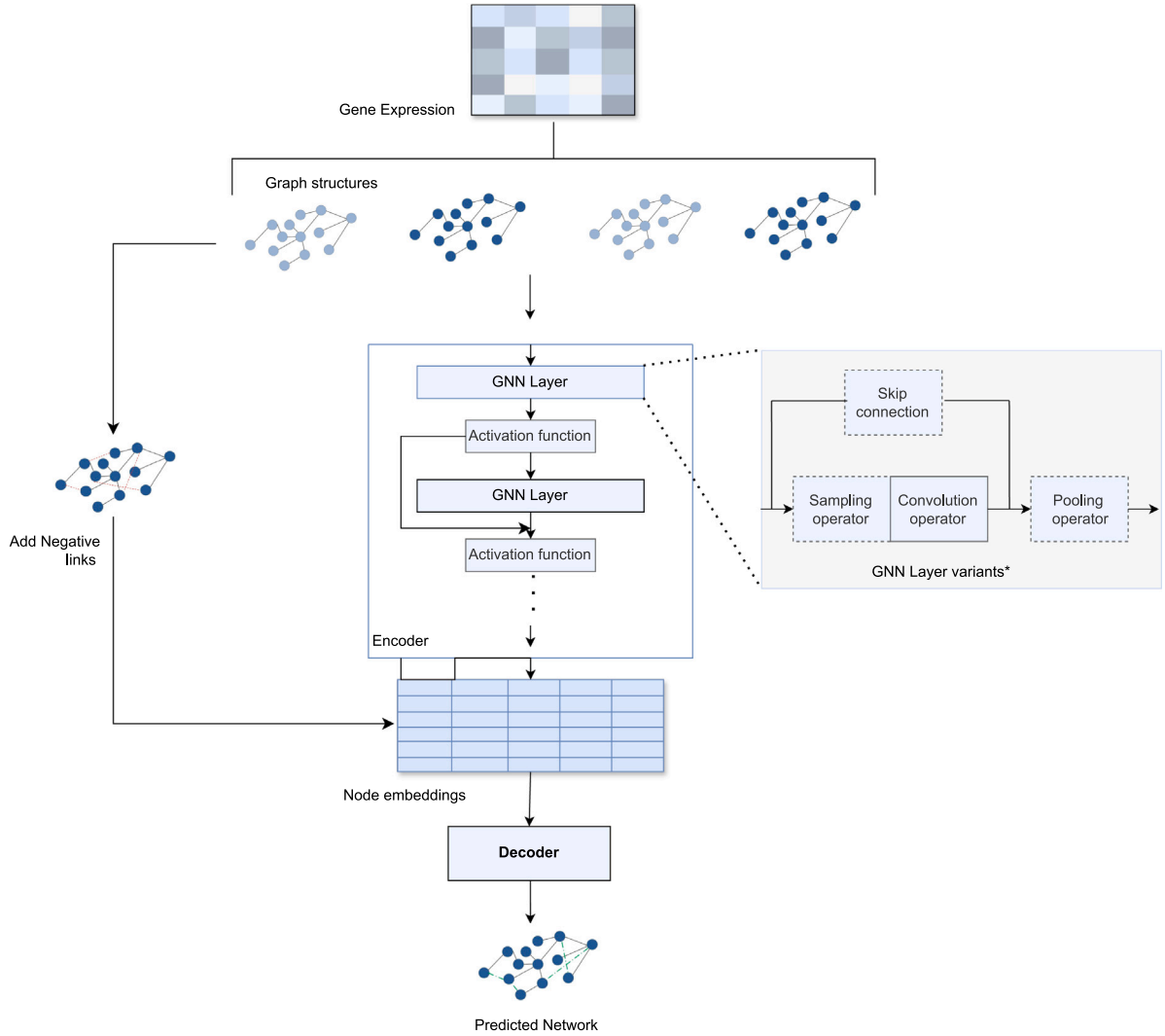


Fig. 2. General architecture of GNN-based graph autoencoder [36]. The GNN layer representation is adapted from [40]. *GNN Layer variants used are listed in supplementary Table 1. The dashed boxes represent the modules that are present in only some GNN layer variants.

process and model parameter learning. The embedding generation process assumes a trained model, where aggregator function parameters are learned. There are K aggregator functions in total for K message passing layers denoted by $AGGREGATE_k$ $k \in \{1, \dots, K\}$ which pass information from one layer to the next. As input graph G and its node features are provided, following which in a minibatch fashion each node v aggregates the representation of nodes in its immediate neighborhood. h_u^{k-1} represents node u 's representation at layer $k-1$ and the aggregation on immediate neighborhood in h_u^{k-1} , $\forall u \in N(v)$ to single vector $h_{N(v)}^{k-1}$. The node's current representation h_u^{k-1} is then concatenated with $h_{N(v)}^{k-1}$. This vector is then passed through a fully connected layer, followed by σ , a nonlinear activation function, which gives representations for the next step. $AGGREGATOR$ has multiple suggested choices. When considering neighboring nodes, GraphSAGE uses uniform sampling of a set with a consistent size of neighbors, rather than a complete neighborhood to ensure a consistent computational footprint. For learning the parameters of the GraphSAGE, loss functions (Eq. (8)) based on the graph are used on the output representations $z_u, \forall u \in V$. Parameters of the $AGGREGATOR$ function and weight matrices are also tuned the same way [47].

$$L_G(z_u) = -\log(\sigma(z_u^T z_v)) - Q \cdot E_{v_n \sim P_n(v)} \log(\sigma(-z_u^T z_{v_n})) \quad (8)$$

where v denotes the neighbor node of u that gets selected under the fixed size criteria mentioned above, P_n is the negative sampling distribution, and Q , is the number of negative samples.

Attention-based spatial methods differentiate the importance of neighboring nodes through varying weights. An example is the famous GAT (Graph Attention Network) which applies an attention mechanism to propagation step [48]. GAT performs self-attention on the vertices and computes the attention coefficients (Eq. (9)).

$$e_{ij} = a(W h_i, W h_j) \quad (9)$$

where $a : R^{F^k} \times R^{F^k} \rightarrow R$ is the shared attentional mechanism. F^k denotes the cardinality of node features at layer k . e_{ij} shows the importance of the node j 's features to node i . This way every node shares information with one another. For the graph scenario, e_{ij} is computed only for $j \in N_i$, first-order neighbors. The coefficients are normalized using Eq. (10) [48] to make them comparable across nodes.

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \quad (10)$$

A single-layer feedforward neural network using the weight vector followed by LeakyReLU nonlinearity is used for a . The fully expanded

equation for coefficient computation can be written as Eq. (11) [48].

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}h_i \parallel \mathbf{W}h_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}h_i \parallel \mathbf{W}h_k]))} \quad (11)$$

where T stands for transposition and \parallel for concatenation operator. The α_{ij} is used for the computation of output features of nodes using a linear combination of input features. The work is also extended using a multi-head attention mechanism.

Skip connections were used as they improved the performance on deeper models where noise propagation and over-smoothing occur.

3.3.2. Sampling modules

For each node, an aggregation operation is performed to aggregate messages from neighbors, but as the number of GNN layers increases the number of neighbors also increases exponentially, which also causes issues in storing neighborhood information. Node sampling reduces neighborhood node size as in GraphSAGE [47].

3.3.3. Pooling modules

Convolutional layers are typically followed by pooling layers to extract higher-level, abstract features. There are two categories under it namely direct and hierarchical [40]. Direct pooling modules learn graph-level representations directly from nodes using various node selection strategies, GraphConv [49] is an example. Hierarchical pooling modules used in ChebConv [44] and LEConv [50] follow a layer-by-layer hierarchical pattern to learn graph representations, often employing graph coarsening algorithms to merge nodes based on spectral clustering techniques.

3.3.4. Other categories

Apart from these some variants of GNNs are often applied on special networks. Hypergraph Convolution [51] is one such method. Unlike regular graphs, Hypergraphs represented by $G = (V, E, W_e)$ can have edges that connect two or more vertices. The edges are assigned a weight $w \in W_e$. HGNN [52] is a technique using Hypergraph convolution to process higher-order interaction between nodes.

Hypergraph G is represented by an incidence matrix defined as $H \in R^{|V| \times |E|}$. If vertex v_i is connected by $e \in E$, $H_{ie} = 1$, else 0. Hyperconvolution is defined as

$$x_i^{(k+1)} = \sigma \left(\sum_{j=1}^{|V|} \sum_{e=1}^{|E|} H_{ie} H_{je} W_e x_j^{(k)} \mathbf{P} \right) \quad (12)$$

where $x_i^{(k)}$ denotes embedding of i th node at k th layer, σ represents a sigmoid function and P is the weight matrix between layers [52]. Eq. (12) can be represented in matrix form as

$$\mathbf{X}^{(k+1)} = \sigma(\mathbf{H}\mathbf{W}\mathbf{H}^T \mathbf{X}^{(k)} \mathbf{P}) \quad (13)$$

To prevent the risk of exploding/vanishing gradients symmetric normalization or row-normalization is performed. The attention learning module is also applied to enrich the incidence matrix H .

There are works focussed on graphs with signed edges, SGCN [53] uses balance theory to predict links between positive and negative edges. Zhou et al. [40] provides a comprehensive overview of GNNs, including their design pipeline and various modules.

3.4. GNN design space for GRN inference

A total of 1,274,796 different designs were constructed over 19 network datasets from DREAM3, DREAM4, and DREAM5 challenges. In this work, a good number of architectures are covered to shed more light on the design space alongside the design chosen to discover a successful GNN model for GRN inference. The work aims to provide a quick look into the combinations that help in making good design choices. **The train, validation, and test splits were in the ratio of 50%, 20%, and 30%.**

A comprehensive set of experiments was conducted using a total of 23 types of convolutional layers, 8 activation functions, 5 aggregation operators, and 3 decoder types, with a fixed layer count of 2, on the DREAM3 size 50 dataset. Subsequently, based on the results analysis, a refined set comprising 4 convolutional layer types, 3 activation functions, 2 aggregators, and 2 decoders was applied to the DREAM4 and DREAM5 datasets.

To gauge the impact of dropouts, rates of 0.2 and 0.5 were employed on the DREAM4 and DREAM5 datasets. An additional analysis involved skip connections, linking the previous layer to the subsequent layer. The number of layers utilized for DREAM4, and DREAM5 datasets varied between 2 and 5, with a deliberate limitation to prevent over-smoothing effects. The study utilized the Adam optimizer, employed BCEWithLogitsLoss as the loss function, and set the learning rate to 0.01. As the dataset size was not affecting computation batching was not performed.

The design dimensions applied for each dataset are provided in the supplementary Table 1.

3.5. Performance metrics

The edges of the constructed dataset are compared with the ground truth to evaluate the model. A Receiver Operator Characteristic curve (ROC) illustrates the relationship between the True Positive Rate (also known as recall) and False Positive Rate (FPR) across different threshold values, whereas a Precision-Recall curve (PRC) depicts the trade-off between precision and recall for varying threshold values. The AUROC and AUPR values correspond to the Area under ROC and the Area under PRC. The calculation of Recall, FPR, and Precision are shown in Eq. (14).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (14a)$$

$$\text{FPR} = \frac{FP}{FP + TN} \quad (14b)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (14c)$$

where TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative. True Positives (TP) represent the edges that originally existed in the network and were correctly predicted by the model. True Negatives (TN) correspond to edges that never existed in the original network and are correctly absent in our predicted network. False Positives (FP) are edges that did not exist in the real network but were erroneously predicted by our model. False Negatives (FN) are edges present in the real network but not identified by our model. Recall and precision often exhibit an inverse relationship, higher recall typically results in lower precision and a higher FPR, while higher precision generally means lower recall but a lower FPR. These trade-offs can be visualized with AUROC and AUPR curves, which summarize performance across different thresholds. Higher AUPR indicates a model's effectiveness in capturing positive interactions, while higher AUROC reflects better overall discrimination between positive and negative interactions.

4. Results and discussion

4.1. DREAM3 datasets

The DREAM3 datasets consisted of five size 50 gene network expression datasets. Size 50 gene networks were utilized to shortlist the convolution layers from 23 to 4. The AUROC and AUPR details of the size 50 gene set are provided in Fig. 3. In the DREAM3 size 50 dataset, a total of 11,041 combinations were tested for each network. The results of this experiment served as a primary filter for subsequent stages of the study. Each combination was run 10 times, and the results were averaged. Each design dimension value was then averaged over the rest of the dimensions to evaluate the overall performance.

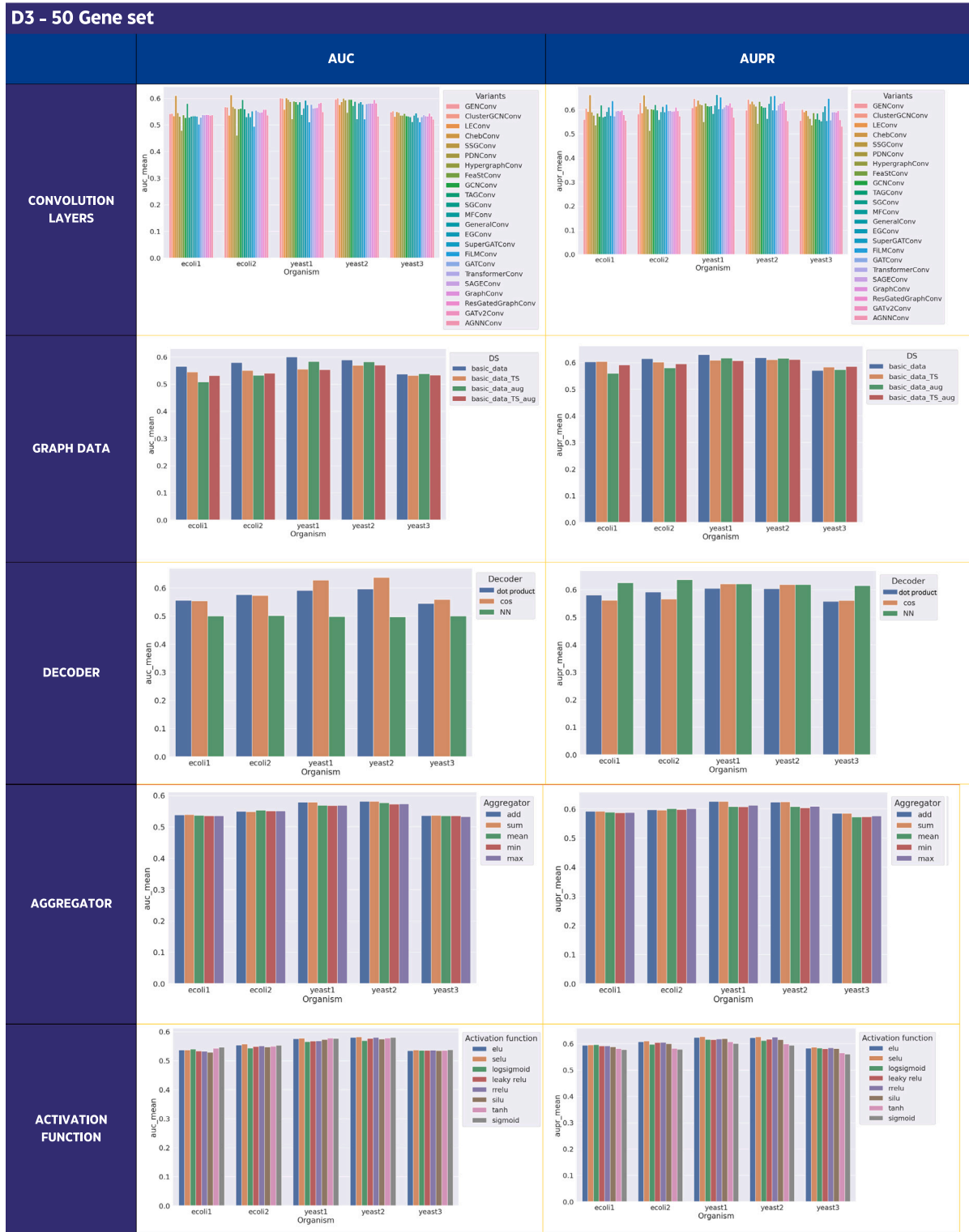


Fig. 3. AUROC and AUPR results in DREAM3 size 50 dataset for design dimensions convolution layers, graph data, decoder, aggregator, and activation function. auc_mean denotes the mean AUROC value over 10 runs, and AUPR_mean denotes the mean AUPR over 10 runs.

Among the 23 convolution layer variants, four of them stood out and were chosen: ChebConv [44], Hypergraph [51], ClusterGCN [54] and SSGConv [55] convolutional layers. Among the graph structures used, those involving time series data seemed to perform lower than the stationary ones. Hence, for further experiments, only static datasets were used. However, for evaluating the current model against state-of-the-art methods, time series datasets are incorporated. Among decoders, dot

product and cosine scores were retained for the next experiment. The neural network-based decoder also shows promising results. Among aggregators, add and sum perform slightly better than the others.

The performance of the activation function varied based on the convolution layer used alongside. The activation functions SiLU, tanh, and sigmoid were selected due to their improved performance when paired with the ChebConv and Hypergraph convolution layers.

Table 3

Comparison of the proposed model against existing techniques using AUROC and AUPR scores in the DREAM4 dataset. GB — Gradient Boost, AB — AdaBoost CC — ChebConv, HG — Hypergraph. The performance measures of baseline methods are collected from GRADIS [56] and GreyNet papers [7].

Method	Net 1		Net 2		Net 3		Net 4		Net 5	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
GNN(CC)	0.589	0.618	0.638	0.654	0.56	0.577	0.617	0.621	0.589	0.613
GNN(HG)	0.658	0.661	0.729	0.745	0.597	0.613	0.553	0.599	0.662	0.655
GreyNet	0.822	0.258	0.725	0.16	0.771	0.267	0.731	0.205	0.789	0.221
BTNET(GB)	0.776	0.186	0.694	0.113	0.759	0.235	0.723	0.143	0.758	0.165
BTNET(AB)	0.776	0.207	0.699	0.116	0.77	0.224	0.74	0.158	0.78	0.169
SWING-RF	0.793	0.192	0.723	0.116	0.759	0.214	0.742	0.193	0.775	0.16
SWING-Dionesus	0.772	0.124	0.7	0.095	0.709	0.194	0.727	0.187	0.771	0.143
BiXGBoost	0.744	0.138	0.682	0.075	0.716	0.119	0.702	0.106	0.728	0.09
GENIE3-time	0.79	0.167	0.711	0.103	0.767	0.215	0.742	0.152	0.786	0.146
Jump3	0.724	0.099	0.623	0.057	0.696	0.077	0.662	0.072	0.696	0.074
TIGRESS	0.715	0.054	0.532	0.037	0.483	0.018	0.467	0.018	0.521	0.022

Table 4

Comparison of the proposed model against existing techniques using AUROC and AUPR scores in the DREAM5 dataset. WOC — Wisdom of crowds, CC — ChebConv, HG — Hypergraph. The performance measures of baseline methods are collected from GRADIS [56] and GRGNN [24] papers.

	TIGRESS	Mrnet	GENIE3	WOC	GRGNN	GRADIS	GNN(CC)	GNN(HG)
InSilico	0.74	0.74	0.82	0.81	–	0.85	0.938	0.895
E. coli	0.59	0.59	0.69	0.69	0.91	0.94	0.917	0.966
S. cerevisiae	0.52	0.52	0.54	0.54	0.90	0.96	0.968	0.98

4.2. DREAM4 and DREAM5 datasets

For DREAM4 and DREAM5, two additional design choices on the number of layers and the number of epochs were added. In graph data, as depicted in Figs. 4 and 7, augmented data has slightly better performance than the basic model. Among decoders, the dot product decoder outperforms the cos decoder, while the aggregator has comparable results. Increasing the number of layers shows an increase in the performance of the Hypergraph but causes an eventual decrease for the other convolution layers. The other three convolution layers seem to perform best with three layers. As indicated in Figs. 5, 6, there is a minor improvement in performance with the initial increase in epoch count, which then soon reaches saturation. The *tanh* and *sigmoid* functions achieve better results for both DREAM4 and DREAM5 datasets.

Based on the results of DREAM4 and DREAM5 datasets, ChebConv and Hypergraph convolution layers performed the best and were further experimented on to understand the effect of skip connection, dropout, and hidden dimension choices. The results are provided in supplementary Figs. 1 and 2. For the DREAM4 challenge, adding skip connection, dropout, and increasing hidden dimension is not favorable when it comes to the ChebConv network. In Hypergraph, skip connection and increasing the hidden dimension show positive results. Dropout does not increase the performance in the Hypergraph from the DREAM4 challenge.

On the DREAM5 challenge dataset, skip connection improves the performance of ChebConv but does not improve the performance of Hypergraph. For ChebConv, dropouts decrease the performance in the DREAM5 challenge and do not show any notable difference for Hypergraph Convolution. Increasing the hidden dimension positively improves the results of ChebConv layers and slightly improves the performance of Hypergraph.

Skip connection has a mixed result; based on the dataset and convolution layer type used, the performance improves or drops. Dropouts are typically introduced to enhance the robustness and generalizability of features while preventing overfitting. In this case, dropouts seem to have low performance as the number of nodes is not that high, affecting the information passing. Hidden dimension variation seems to have more impact on Hypergraphs than ChebConv in DREAM4 datasets, and on larger DREAM5 datasets, it shows performance improvement in both convolution layer types.

4.3. Comparison with other related work

4.3.1. DREAM4

The comparison of the two best-performing models in GNN with state-of-the-art techniques is provided in Table 3.

In DREAM4 challenge a significant improvement can be seen in the AUPR scores using GNN methods. However, the AUROC values do not show the same improvement. GreyNet shows higher accuracy values in 3 networks compared to that of the GNN-based approach.

4.3.2. DREAM5

The use of both the ChebConv and the Hypergraph approaches on InSilico and real networks of E. coli and S.cerevisiae show significant improvement (Figs. 4 and 7). The AUROC results for the InSilico and real datasets of the DREAM5 challenge are compared against other works and results are as shown in Table 4. ChebConv utilizes techniques from spectral graph theory, particularly the graph Laplacian matrix, to define graph convolutions. Chebyshev polynomials are used to approximate localized spectral filters, to capture local graph features that are valuable for link prediction tasks. This approximation makes it well-suited for practical applications, including those involving large and deep networks. This adaptability contributes to its effectiveness with both simulated and real datasets. Hypergraphs extend the capabilities of traditional graphs by accommodating more intricate relationships among nodes. Hyperedges can connect multiple nodes simultaneously, providing greater expressiveness for modeling complex data relationships. These convolutional layers excel in capturing higher-order dependencies among nodes, which proves valuable when relationships involve multiple interconnected nodes. GRNs are sparse graphs due to the low count of edges against the non-edges. Hypergraphs demonstrate proficiency in managing sparse data by effectively capturing higher-order relationships [51].

In social network settings and e-commerce recommendation systems, hypergraphs have proven effective in addressing challenges stemming from sparse network data [57,58]. This can be attributed to the superior performance observed in GRNs when employing hypergraph convolution layers. In semi-supervised learning scenarios, Hypergraph convolutional layers leverage both node features and hyperedge information to make predictions. Compared to GRGNN, a GNN-based model, the proposed technique demonstrates superior results. However, it is worth noting that GRGNN was designed for inductive inference of GRNs. Nonetheless, it serves as a valuable comparison technique, given that both approaches utilize GNNs to address the inference problem.

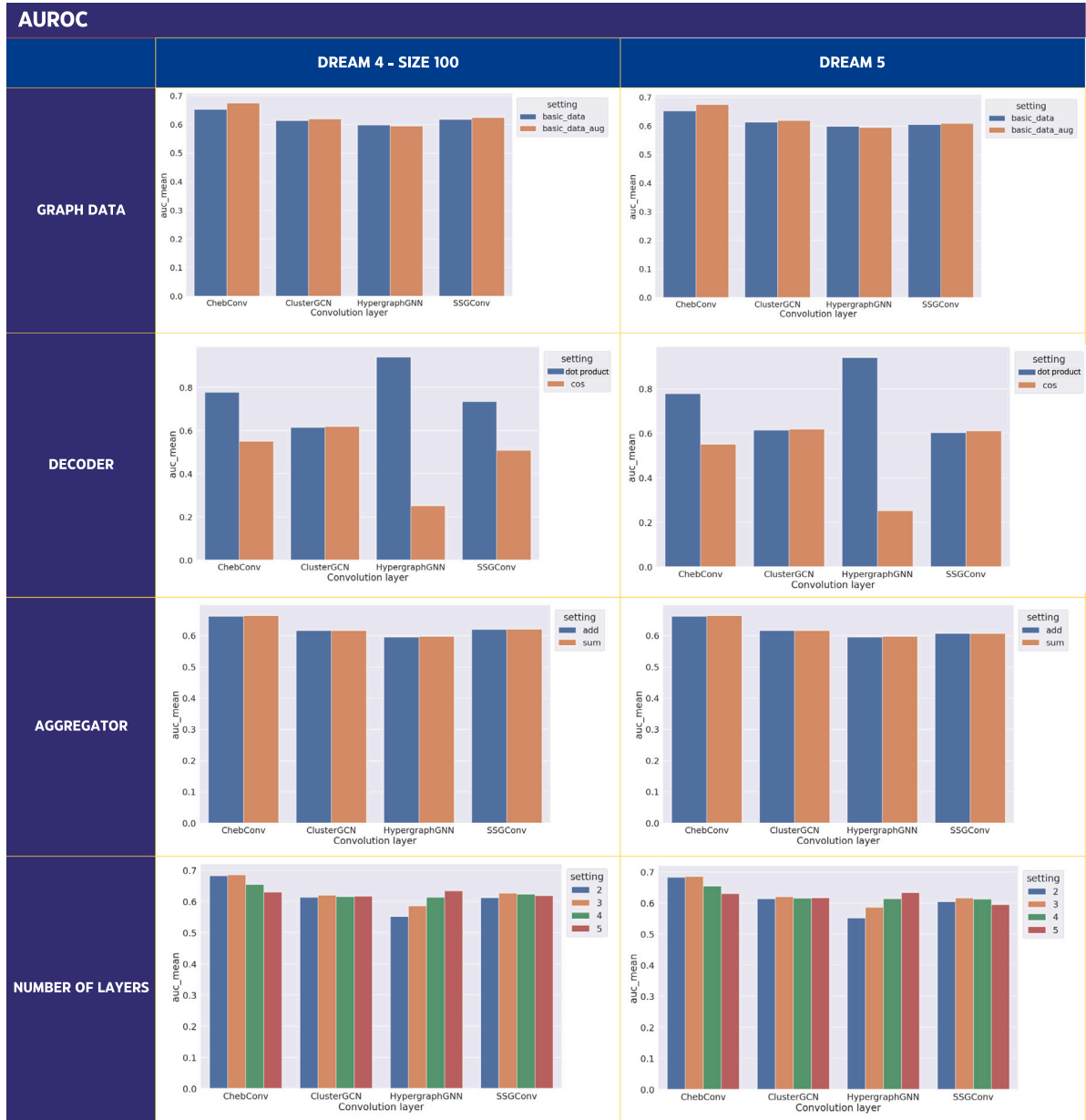


Fig. 4. AUROC results in DREAM4 size 100 dataset and DREAM5 dataset for design dimensions graph data, decoder, aggregator, and number of layers. auc_mean denotes the mean AUROC value over 10 runs, setting represents the values for the parameter under consideration.

5. Conclusion

GRN inference is an important research area that holds great hopes for understanding the regular workings of the human body and disease pathology. GRNs can uncover disease mechanisms, and shortlist disease-causing genes by highlighting their regulatory relations with other genes [59]. We leveraged the DREAM3, DREAM4, and DREAM5 gene expression datasets to derive our findings and compared them with state-of-the-art techniques. Notably, our analysis revealed that GNN methods outperformed other approaches in terms of performance scores.

Here are the conclusions drawn from the experiments above:

1. The convolution operator is a crucial component of GNNs, and the choices made in selecting the appropriate operator can significantly impact the results depending on the specific task at hand. In this research, two specific convolution layers, namely ChebConv and Hypergraph, demonstrated outstanding performance in the context of GRN inference.

2. Eliminating features through dropout does not seem to provide significant benefits. This might be because GNNs are already robust to noise and outliers.
3. Regarding aggregation methods based on the results of the DREAM3 challenge, add and sum shows promise.
4. The impact of skip connections varies depending on the dataset used, with considerable performance gains observed for ChebConv on the DREAM5 dataset. Skip connections are most effective in deep networks with many layers; however, in shallow networks, their impact on model performance may be less significant.
5. Increasing the number of layers has a positive impact on Hypergraph and ChebConv models but negatively affects SSGConv, ClusterGCN, and ChebConv models, likely due to over-smoothing issues and overfitting.
6. Increasing the number of epochs initially improves performance but soon reaches saturation, indicating the possibility of overfitting.



Fig. 5. AUROC results in DREAM4 size 100 dataset and DREAM5 dataset for design dimensions number of epochs, activation function. auc_mean denotes the mean AUROC value over 10 runs, setting represents the values for the parameter under consideration.

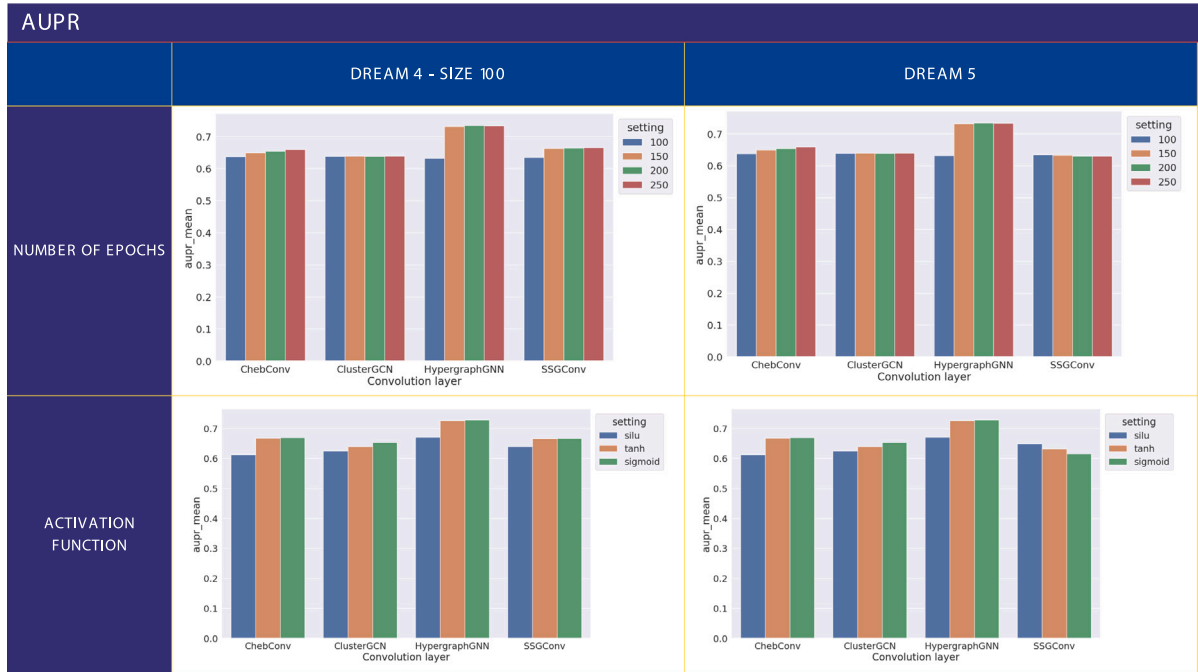


Fig. 6. AUPR results in DREAM4 size 100 dataset and DREAM5 dataset for design dimensions number of epochs, activation function. aupr_mean denotes the mean AUPR value over 10 runs, setting represents the values for the parameter under consideration.

7. No single activation function consistently outperformed others in DREAM3 size 50 datasets when averaged across various parameters. The ideal choice of activation function for optimal performance depends on the specific GNN layer and the characteristics of the dataset in question.
8. The dot product decoder consistently outperforms other decoders, while the neural network-based decoder warrants further investigation.

9. Incorporating time series information into graph data generally reduces overall GNN performance, suggesting the need for alternative representations.
10. The model shows good results in DREAM5, which is noteworthy as it includes two real gene expression datasets.

In addition to these observations, our study did not explore learning configuration parameters, Batch Normalization, interlayer designs, or various skip connection variants. Future research could investigate the

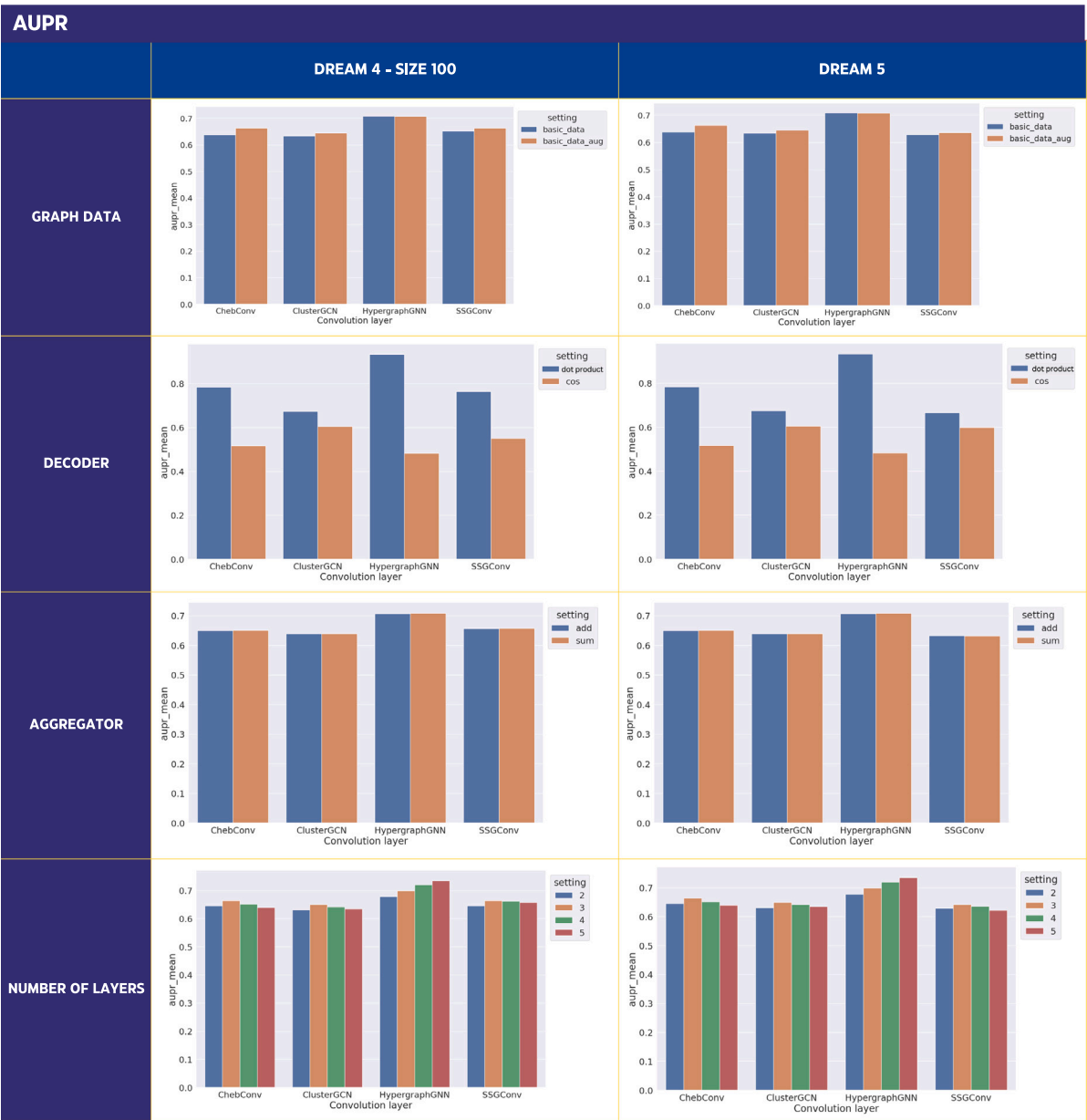


Fig. 7. AUPR results in DREAM4 size 100 dataset and DREAM5 dataset for design dimensions graph data, decoder, aggregator, and number of layers. aupr_mean denotes the mean AUPR value over 10 runs, setting represents the values for the parameter under consideration.

impact of these dimensions and explore alternative decoder variants and graph structures to represent biological data better.

CRediT authorship contribution statement

Emma Paul M.: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. Jereesh A.S.: Writing – review & editing, Supervision. G. Santhosh Kumar: Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the github link to code and the datasets used.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.asoc.2024.111899>.

References

[1] L.E. Chai, S.K. Loh, S.T. Low, M.S. Mohamad, S. Deris, Z. Zakaria, A review on the computational approaches for gene regulatory network construction, *Comput. Biol. Med.* 48 (2014) 55–65.

[2] P. Langfelder, S. Horvath, WGCNA: an r package for weighted correlation network analysis, *BMC Bioinform.* 9 (2008) 1–13.

[3] A.K. Tan, M.S. Mohamad, Using Bayesian networks to construct gene regulatory networks from microarray data, *J. Teknol.* (2012) 1–6.

[4] M. Hecker, S. Lambeck, S. Toepfer, E. Van Someren, R. Guthke, Gene regulatory network inference: data integration in dynamic models—a review, *Biosystems* 96 (1) (2009) 86–103.

[5] P.E. Meyer, K. Kontos, F. Lafitte, G. Bontempi, Information-theoretic inference of large transcriptional regulatory networks, *EURASIP J. Bioinform. Syst. Biol.* 2007 (2007) 1–9.

- [6] H. Richards, Y. Wang, T. Si, H. Zhang, H. Gong, Intelligent learning and verification of biological networks, in: *Advances in Artificial Intelligence, Computation, and Data Science: for Medicine and Life Science*, Springer, 2021, pp. 3–28.
- [7] G. Chen, Z.-P. Liu, Inferring causal gene regulatory network via GreyNet: From dynamic grey association to causation, *Front. Bioeng. Biotechnol.* 10 (2022) 954610.
- [8] J.D. Finkle, J.J. Wu, N. Bagheri, Windowed granger causal inference strategy improves discovery of gene regulatory networks, *Proc. Natl. Acad. Sci.* 115 (9) (2018) 2252–2257.
- [9] M. Paul, Emma, A.S. Jereesh, G.S. Kumar, in: S. Philip (Ed.), *Proceedings of the International Conference on Artificial Intelligence & Software Engineering, ICAISE 2023 Kochi*, Directorate of Public Relation and Publications, Cochin University of Science And Technology, 2023, pp. 23–28.
- [10] P. Patil, M. Vaida, Learning gene regulatory networks using graph granger causality, in: *Proceedings of 14th International Conference*, Vol. 83, 2022, pp. 10–19.
- [11] D. Marbach, J.C. Costello, R. Küffner, N.M. Vega, R.J. Prill, D.M. Camacho, K.R. Allison, M. Kellis, J.J. Collins, et al., Wisdom of crowds for robust gene network inference, *Nat. Methods* 9 (8) (2012) 796–804.
- [12] S. Mandal, G. Saha, R.K. Pal, Neural network based gene regulatory network reconstruction, in: *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology, C3IT, IEEE*, 2015, pp. 1–5.
- [13] D. MacLean, A convolutional neural network for predicting transcriptional regulators of genes in arabidopsis transcriptome data reveals classification based on positive regulatory interactions, 2019, *bioRxiv*, 618926.
- [14] H. Shrivastava, Reconstruction of gene regulatory networks using sparse graph recovery models, 2023, *bioRxiv*, 2023-2004.
- [15] D. Liben-Nowell, J. Kleinberg, The link prediction problem for social networks, in: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, 2003, pp. 556–559.
- [16] L. Lü, T. Zhou, Link prediction in complex networks: A survey, *Phys. A* 390 (6) (2011) 1150–1170.
- [17] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, *Science* 286 (5439) (1999) 509–512.
- [18] T. Zhou, L. Lü, Y.-C. Zhang, Predicting missing links via local information, *Eur. Phys. J. B* 71 (2009) 623–630.
- [19] L. Page, Method for node ranking in a linked database, 1997, USA Patent 6.
- [20] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine (reprint from computer networks and isdn systems, vol 30, pg 107-117, 1998), *Comput. Netw.* 56 (18) (2012) 3825–3833.
- [21] M. Zhang, Y. Chen, Weisfeiler-lehman neural machine for link prediction, in: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 575–583.
- [22] I.A. Kovács, K. Luck, K. Spirohn, Y. Wang, C. Pollis, S. Schlabach, W. Bian, D.-K. Kim, N. Kishore, T. Hao, et al., Network-based prediction of protein interactions, *Nat. Commun.* 10 (1) (2019) 1240.
- [23] M. Zhang, Y. Chen, Link prediction based on graph neural networks, *Adv. Neural Inf. Process. Syst.* 31 (2018).
- [24] J. Wang, A. Ma, Q. Ma, D. Xu, T. Joshi, Inductive inference of gene regulatory network using supervised and semi-supervised graph neural networks, *Comput. Struct. Biotechnol. J.* 18 (2020) 3335–3343.
- [25] T. Hamaguchi, H. Oiwa, M. Shimbo, Y. Matsumoto, Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach, 2017, *arXiv preprint arXiv:1706.05674*.
- [26] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, *arXiv preprint arXiv:1609.02907*.
- [27] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, 2017, *arXiv preprint arXiv:1709.05584*.
- [28] K. Feng, H. Jiang, C. Yin, H. Sun, Gene regulatory network inference based on causal discovery integrating with graph neural network, *Quant. Biol.* 11 (4) (2023) 434–450.
- [29] H. Zhang, X. An, Q. He, Y. Yao, F.-L. Fan, Y. Teng, Quadratic graph attention network (Q-GAT) for robust construction of gene regulatory networks, 2023, *arXiv preprint arXiv:2303.14193*.
- [30] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proc. IEEE* 104 (1) (2015) 11–33.
- [31] T.-W. Chang, Binding of cells to matrixes of distinct antibodies coated on solid surface, *J. Immunol. Methods* 65 (1–2) (1983) 217–223.
- [32] D. Marbach, R.J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, G. Stolovitzky, Revealing strengths and weaknesses of methods for gene network inference, *Proc. Natl. Acad. Sci.* 107 (14) (2010) 6286–6291.
- [33] D. Marbach, T. Schaffter, C. Mattiussi, D. Floreano, Generating realistic in silico gene networks for performance assessment of reverse engineering methods, *J. Comput. Biol.* 16 (2) (2009) 229–239.
- [34] R.J. Prill, D. Marbach, J. Saez-Rodriguez, P.K. Sorger, L.G. Alexopoulos, X. Xue, N.D. Clarke, G. Altan-Bonnet, G. Stolovitzky, Towards a rigorous assessment of systems biology models: the DREAM3 challenges, *PLoS One* 5 (2) (2010) e9202.
- [35] W.Z. Ouma, K. Pogacar, E. Grotewold, Topological and statistical analyses of gene regulatory networks reveal unifying yet quantitatively different emergent properties, *PLoS Comput. Biol.* 14 (4) (2018) e1006098.
- [36] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, *arXiv preprint arXiv:1611.07308*.
- [37] M. Zhao, W. He, J. Tang, Q. Zou, F. Guo, A comprehensive overview and critical evaluation of gene regulatory network inference technologies, *Brief. Bioinform.* 22 (5) (2021) bbab009.
- [38] D. Mercatelli, L. Scalambra, L. Triboli, F. Ray, F.M. Giorgi, Gene regulatory network inference resources: A practical overview, *Biochim. Biophys. Acta (BBA)-Gene Regul. Mech.* 1863 (6) (2020) 194430.
- [39] M. Fey, J.E. Lenssen, Fast graph representation learning with PyTorch Geometric, in: *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [40] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, *AI Open* 1 (2020) 57–81.
- [41] W.L. Hamilton, Graph Representation Learning, Morgan & Claypool Publishers, 2020.
- [42] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, *IEEE Signal Process. Mag.* 30 (3) (2013) 83–98.
- [43] S. Mallat, A Wavelet Tour of Signal Processing, Elsevier, 1999.
- [44] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, *Adv. Neural Inf. Process. Syst.* 29 (2016).
- [45] D.K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, *Appl. Comput. Harmon. Anal.* 30 (2) (2011) 129–150.
- [46] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, 2018.
- [47] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, *Adv. Neural Inf. Process. Syst.* 30 (2017).
- [48] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, *arXiv preprint arXiv:1710.10903*.
- [49] C. Morris, M. Ritzert, M. Fey, W.L. Hamilton, J.E. Lenssen, G. Rattan, M. Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 4602–4609.
- [50] E. Ranjan, S. Sanyal, P. Talukdar, Asap: Adaptive structure aware pooling for learning hierarchical graph representations, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, 2020, pp. 5470–5477.
- [51] S. Bai, F. Zhang, P.H. Torr, Hypergraph convolution and hypergraph attention, *Pattern Recognit.* 110 (2021) 107637.
- [52] Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 3558–3565.
- [53] T. Derr, Y. Ma, J. Tang, Signed graph convolutional networks, in: *2018 IEEE International Conference on Data Mining, ICDM, IEEE*, 2018, pp. 929–934.
- [54] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.
- [55] H. Zhu, P. Koniusz, Simple spectral graph convolution, in: *International Conference on Learning Representations*, 2020.
- [56] Z. Razaghi-Moghadam, Z. Nikoloski, Supervised learning of gene-regulatory networks based on graph distance profiles of transcriptomics data, *NPJ Syst. Biol. Appl.* 6 (1) (2020) 21.
- [57] Y. Guan, X. Sun, Y. Sun, Sparse relation prediction based on hypergraph neural networks in online social networks, *World Wide Web* 26 (1) (2023) 7–31.
- [58] Y. Han, E.W. Huang, W. Zheng, N. Rao, Z. Wang, K. Subbian, Search behavior prediction: A hypergraph perspective, in: *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 697–705.
- [59] F. Emmert-Streib, M. Dehmer, B. Haibe-Kains, Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks, *Front. Cell Dev. Biol.* 2 (2014) 38.