Complex & Intelligent Systems (2024) 10:111–128
https://doi.org/10.1007/s40747-023-01068-6

ORIGINAL ARTICLE

Hierarchical graph neural network with subgraph perturbations for key gene cluster discovery in cancer staging

Wenju Hou[1] · Yan Wang[1,3] · Ziqi Zhao[2] · Yizhi Cong[3] · Wei Pang[4] · Yuan Tian[3]

Received: 21 November 2022 / Accepted: 1 April 2023 / Published online: 17 July 2023

Abstract

Analyzing highly individual-specific genomic data to understand genetic interactions in cancer development is still challenging, with significant implications for the discovery of individual biomarkers as well as personalized medicine. With the rapid development of deep learning, graph neural networks (GNNs) have been employed to analyze a wide range of biomolecular networks. However, many neural networks are limited to black box models, which are only capable of making predictions, and they are often challenged to provide reliable biological and clinical insights. In this research, for sample-specific networks, a novel end-to-end hierarchical graph neural network with interpretable modules is proposed, which learns structural features at multiple scales and incorporates a soft mask layer in extracting subgraphs that contribute to classification. The perturbations caused by the input graphs' deductions are used to evaluate key gene clusters, and the samples are then grouped into classes to produce both sample- and stage-level explanations. Experiments on four gene expression datasets from The Cancer Genome Atlas (TCGA) show that the proposed model not only rivals the advanced GNN methods in cancer staging but also identifies key gene clusters that have a great impact on classification confidence, providing potential targets for personalized medicine.

Keywords Sample-specific networks · Graph convolution networks · Graph pooling · Cancer staging · Significant subgraph extraction

✉ Yan Wang
wy6868@jlu.edu.cn

Wenju Hou
houwj20@mails.jlu.edu.cn

Ziqi Zhao
zqzhao21@mails.jlu.edu.cn

Yizhi Cong
congyz20@mails.jlu.edu.cn

Wei Pang
w.pang@hw.ac.uk

Yuan Tian
yuantian@jlu.edu.cn

1 Key Laboratory of Symbol Computation and Knowledge Engineering, Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun 130012, China

2 College of Software, Jilin University, Changchun 130012, China

3 School of Artificial Intelligence, Jilin University, Changchun 130012, China

4 School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH144AS, UK

Introduction

Cancer is a complex, dynamic, and progressive process involving a variety of gene-environment interactions, and it is largely driven by genetic changes. According to recent statistics, the global burden of cancer incidence and mortality is rapidly growing; for some highly frequent malignancies, such as lung cancer, patients' 5-year survival after diagnosis is barely 10% to 20% in most countries [1]. Unfortunately, it is still difficult to comprehend how cancer develops, particularly in its early stages. The tumor/node/metastasis (TNM) system, the most common cancer staging system which classifies and characterizes cancer development into several stages, has been widely utilized to guide further investigation and understanding of cancer development at both molecular and clinical levels [2, 3].

Personalized medicine is a promising and rapidly developing therapy option, thanks to the novel bioinformatics tools and a better understanding of tumor biology [4–6]. The processes of gene expression and gene regulatory networks, as complex systems in the real world, can be well described and modeled using mathematical equations [7]. The practice of

treating lung cancer, for instance, has changed to a hallmark of personalized medicine, with subsets of patients treated according to the genetic alterations of their tumor and the status of programmed death ligand-1 (PD-L1) instead of the empirical application of cytotoxic therapy based on a doctor's preference [5]. However, due to the vast genetic heterogeneity of tumor cells between individuals with the same type of cancer and even within individual tumors, the discovery of both more biomarkers for personalized treatment of individual tumors and commonalities among tumors of the same type and stage has become more critical and necessary [6].

A single-sample network or sample-specific network (SSN) is a biomolecular network constructed from single-sample data and a reference dataset, and it is used to characterize an individual's specific disease state [8–10]. In contrast to analyzing individual molecular markers (e.g., genes, metabolites, or proteins), biomolecular networks can capture and model complex biological processes and molecular interactions. In addition, unlike other biological networks that focus on large cohorts, such as gene regulatory networks or co-expression networks, SSNs focus on the information at the individual level and generate feedback on the subject-specific response to pathophysiological stimuli caused by the dysfunction of individual-specific systems [9]. A partial correlation-based single-sample network (P-SSN) is a single-sample network that uses partial correlation coefficients (PTCCs) rather than Pearson's correlation coefficients (PCCs) [11]. The P-SSN approach distinguishes itself by excluding indirect/cascading gene interactions from network construction, hence highlighting direct interactions. Although P-SSN has shown feasibility and practicality in some downstream tasks such as predicting potential driver mutation genes, tumor/normal sample classification, and tumor sample clustering, further extensive applications and analysis of P-SSN still need to be undertaken.

With the development of deep learning methods, deep neural networks have been widely employed in bioinformatics with remarkable success. In particular, graph neural networks (GNNs) have become a promising tool for analyzing biomolecular networks [12, 13]. A graph convolutional network (GCN) is a generalization of convolutional operations from grid data to graph data, and its basic principle is to update node representations by aggregating features from nodes and their neighbors. Because these GCNs are inherently flat, they perform well in node classification and link prediction but fail to handle graph classification directly. As a result, either a global operation for obtaining graph-level representations (readout) or a network structure capable of aggregating information hierarchically is required, the latter being commonly referred to as graph pooling operators. Lastly, a Multilayer Perceptron (MLP) is frequently used to accept graph representations as input and produce classification results. However, interpretability is a common challenge in current deep learning algorithms in biology and medicine. Furthermore, when modeling another complicated system of gene regulatory networks, neural networks, as complex systems, are unavoidably challenged by disturbances, modeling errors, and various uncertainties in the real systems [7, 14–16]. These systems frequently work as "black boxes", preventing practitioners from understanding useful patterns and motifs in data detected by models that reach breakthrough performance [12].

It is still challenging to apply graph neural networks to stage cancer patients and identify the genes that are significant in the classification. First, because there is no natural graph structure in the tabular gene expression data of cancer patients, it is necessary to manually extract the links between the genes. Second, the existing differentiable graph pooling operators consume a significant amount of computing resources, limiting their performance on large and dense networks. Existing graph classification networks are also unsuitable for graph classification and community detection simultaneously. Finally, despite tremendous research into key genes in cancer, it is still not ready to conclude that there is credible supervised information in this problem, which greatly hampers training and interpreting models.

Considering the above, in this research, we proposed a complete workflow and a novel end-to-end graph neural network architecture for classifying early- and late-stage cancers and discovering contributing gene clusters. To address the key challenges discussed above, we introduce novel network construction approaches and improve existing graph neural architectures and graph pooling operators to perform classification and community detection. Figure 1 depicts the overall process. We used the P-SSN method to convert the patients' gene expression data into graphs as input to the graph neural network for retaining the individual specificity of the data and results. A graph classification network with hierarchical pooling serves as the backbone of the proposed graph neural network, with an additional differentiable soft mask layer for learning the contributing subgraph structures. Based on the perturbation, we anticipate a reliable interpretation, i.e., the process of re-passing the classification after removing the significant subgraphs to confirm the impact of the learned structure on the prediction. Then, we investigated the potential biological functions and connections of these structures and observed clues with practical applications. Through the interpretability of cancer staging classification models, we provide a feasible supervised solution to the unsupervised problem of discovering key gene clusters. We conducted systematic experiments on several real-world datasets and compared our proposed model with several state-of-the-art graph neural networks, and the results show that our model outperforms state-of-the-art approaches on real-world datasets selected.

In summary, our novel contributions are as follows:

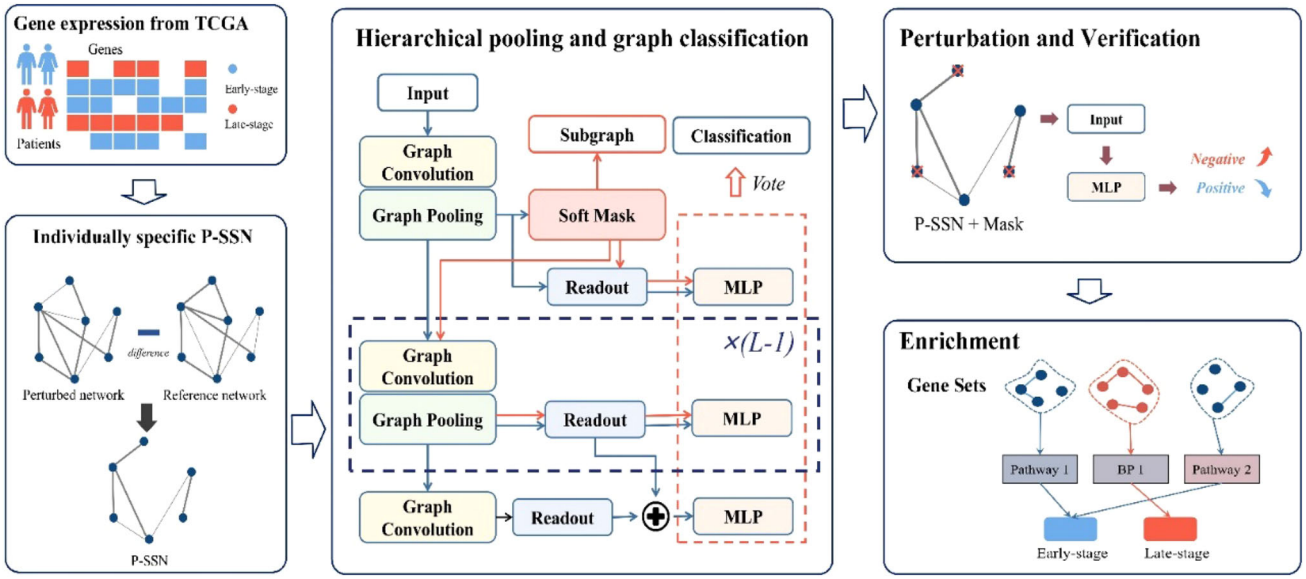

Fig. 1 The workflow of our approach

First, we propose a novel hierarchical graph neural network (HGNN) architecture for cancer staging and the discovery of significant gene clusters with improved graph pooling operators and a soft mask layer. The proposed pooling operator could generate clusters with partially overlapping nodes, thereby avoiding the issue of dense graphs with complete overlap that plagues the standard differentiable graph pooling operator.

Second, we developed a subgraph perturbation strategy and conducted experiments to reconstruct the graphs and annotate the biological functions of genes, and revealed that removing significant subgraphs discovered by the network decreases the confidence of classifier in correct classification, demonstrating the feasibility of the key node discovery strategy.

The rest of the paper is organized as follows: the next section briefly describes related work; the subsequent section presents the process of data collection and pre-processing followed by which the proposed method is formalized;the penultimate section reports and discusses the experimental results; the final section presents our conclusions.

Related work

Graph neural network

As a complex data structure, a graph is made up of nodes and edges (or links) that can be utilized to describe complex systems, such as social networks, protein–protein interaction networks, gene networks, and so on. Graph neural networks have grown into a popular and powerful framework for dealing with graphs. As a typical spatial graph convolution operator, Hamilton et al. present GraphSAGE (SAmple and aggreGatE), a generic framework for obtaining node embeddings that trains a set of aggregator functions that learn to aggregate feature information from a node's local neighborhood [17]. The aggregation functions that are offered include Mean aggregator, LSTM (Long Short-Term Memory) aggregator, and Pooling aggregator. Graph Attention Networks (GATs) introduce an attention mechanism that leverages self-attentive layers to implicitly assign different weights to different nodes in the neighborhood during aggregation [18]. Although these graph convolution approaches have shown good results for node-level tasks, the majority of them cannot be straightforwardly applied to graph-level tasks.

For graph-level tasks such as graph classification, pooling operators are important and useful modules. Global graph pooling (also known as readout) and hierarchical graph pooling are the two graph pooling operators: the former tries to obtain a universal representation of the input graph, while the latter aims to capture enough structural information for node representation [19]. The recent hierarchical graph pooling approaches either coarsen the graph by clustering the nodes [20–23] or sample the nodes after evaluation to reduce the number of nodes [24–27]. Early clustering-based graph pooling often invoked existing graph clustering algorithms like spectral clustering. For example, EigenPooling adopts spectral clustering to obtain the subgraphs with no overlapping nodes to complete the coarsening process, where a subgraph is treated as a supernode and the original graph signal information is translated into the graph signal defined on

the coarsened graph [20]. DiffPool is a differentiable graph pooling operator that can generate hierarchical graph representations by GNNs, learning a soft cluster assignment matrix between nodes and supernodes. GSAPool is a typical top-k pooling operator that evaluates the importance of nodes in multiple ways based on their local structure and feature information and employs the feature fusion method, overcoming the limitation of selecting nodes from a single perspective while improving the pooled graph's feature representation ability [28].

Interpretability in the GNN can be defined as the prediction to the input graph and then sampling a significant subgraph as the explanation of the model prediction [29]. Aiming to answer the question "How does a GNN make a certain decision? Which nodes or features are essential? ", researchers have attempted to explain GNNs via gradient-like importance scores obtained by backpropagating the model outcome to the graph structure, masks or attention scores derived from the masking functions or attention layers, or prediction changes after perturbation of input graphs [30–32]. Wang et al. put forward the concept of multi-grained explainability and proposed ReFine, an explainer with pre-training and fine-tuning techniques for global and local explainability [29]. Yang et al. designed soft-mask GNN layers that identify and remove irrelevant (or noisy) parts of the input graph that are unrelated to the task goal, resulting in subgraphs of arbitrary size and structure and hierarchical graph representations, and the masks can also visualize the structure learned by the model [33].

Detection of cancer driver genes

It has been demonstrated that gene mutations are related to cancer, and cancer is assumed to be caused by an accumulation of genetic mutations. However, only a few genes are identified as cancer driver genes (CDGs), and mutations in these genes contribute to cancer development and progress. Therefore, understanding the molecular mechanism of cancer and designing targeted medicines and diagnostics depend greatly on discovering CDGs. Numerous computational methods, primarily of two types: those based on mutation frequency and those based on networks, have been developed in recent years to unravel CDGs [34, 35]. Somatic mutation is so productive and easy to get that it is practically an essential type of data for identifying driver genes [36–47]. Frequency/recurrence, clustering, functional impact (FI), etc., are all often used features for methods that only utilize mutation data [35, 36, 38, 40, 44, 46]. The mutation frequency-based methods primarily employ statistical significance of higher than the background mutation rate (BMR) to find significantly mutated genes or use ratio metric to detect cancer-driver genes based on the composition of mutation types normalized by the total number of mutations in a gene [44, 46].

Many novel network-based methods for discovering CDGs have been successfully implemented by merging gene network data with various omics data, such as mutations, gene expression, pathways, gene function information, DNA methylation, and so on [36, 47–50]. Most network-based methods include the following stages: (1) building or optimizing gene expression networks, (2) discovering critical nodes/node communities in the constructed networks, and (3) scoring or ranking to identify driver genes [49–52]. NIBNA (node importance-based biological network analysis) constructs a gene network by augmenting gene expression data with knowledge from existing databases, detects communities from the input gene network using the Louvain community detection algorithm, and uses the community structure to compute the importance of nodes in the network [50]. Pham et al. used the network control method to identify cancer drivers as influential nodes in a network that are critical to the control of the system's working, such that removing such a node will require more nodes to control the network [52]. Wei et al. proposed a novel method based on random walk methods to calculate scores for candidate genes and to filter candidate driver genes using mutation data as posterior information [49].

Cancer driver module identification and personalized cancer driver identification have emerged as two further trends in cancer driver identification methods, owing to the heterogeneity of cancer and the complex regulatory interactions between genes [34, 40–43, 45, 51, 52]. Pham et al. define a 'driver gene group' as a group of genes that cooperate to regulate cancer or cancer markers [51]. Zhang et al. develop two mathematical programming models (ComMDP and SpeMDP) to newly identify cancer common and specific driver gene sets from mutation data without relying on prior knowledge [40]. Using individual-based omics data, a novel network integration approach called Bayesian network integration (BNI) is proposed to prioritize personalized driver genes and the corresponding controlled downstream modules [43]. With the advancement of machine learning, more powerful computational techniques for discovering driver genes have been available, and promising progress has been made [37–39, 42–44]. Deep neural networks, particularly graph neural networks applied to network data, have yet to find broad applications. Several studies have been conducted to investigate the feasibility of using graph convolutional networks for node classification in the detection of CDGs [47, 48]. However, there is currently a lack of comprehensive integration of graph neural networks with the general processes of network-based methods for discovering cancer driver genes, such as identifying node communities, evaluating nodes as well as ranking the importance of nodes.

Table 1 A summary of four TCGA cohorts

Cohort	Normal	Early-stage	Late-stage
BRCA	99	823	278
STAD	31	164	188
LUAD	57	422	110
COAD	40	267	154

Data preparation

Data collection

In this study, we used cancer cohorts from The Cancer Genome Atlas (TCGA) project to calculate P-SSN for early-stage and late-stage cancer classification. The cohorts we used include Breast Invasive Carcinoma (BRCA), Stomach Adenocarcinoma (STAD), Lung Adenocarcinoma (LUAD), and Colon Adenocarcinoma (COAD). We downloaded the RNA-seq files and the clinical data for each cohort, where RNA-seq are gene expression data and clinical data provide possible patient cancer staging information. These data are available at https://portal.gdc.cancer.gov.

In total, we obtained 2,633 RNA sequencing files and clinical information for 2,587 patients, and this information is summarized in Table 1. Tumors with Stage I or II annotations were regarded as early-stage cancers (positive samples), whereas those with Stage III or IV annotations were considered late-stage cancers (negative samples). Even though their gene expression profiles were accessible, the patients annotated with Stage X (i.e., 13 patients in BRCA) were not included in our analyses. Furthermore, not all patients have both clinical and gene expression data, thus these instances were also excluded.

Partial correlation-based single-sample networks

The P-SSN algorithm generates graphs from all labeled gene expression data. To construct the reference networks, all normal samples in each cohort are used as reference datasets [11]. The P-SSN calculating procedure is divided into the following steps: (1) building a background network by calculating the PCCs between any two genes based on gene expression in reference samples and keeping significantly correlated (PCC > 0.7) gene pairs as edges, (2) building a reference network by keeping significant edges and excluding non-significant edges from the background network based on the PTCCs of reference samples, (3) combining a new sample with the reference samples, recalculating the PTCCs, and building the perturbed network, (4) calculating the sample-specific PTCCs (sPTCCs) and constructing final P-SSN for a single sample by keeping significant edges and eliminating nonsignificant edges of the sample from the background network based on sPTCCs. By analyzing tumor data from TCGA and single cell data, the effectiveness of P-SSN in predicting DMGs, identifying subtypes, and further classifying single cells was validated [11]. P-SSN, in particular, has great potential in predicting DMGs and biological network biomarkers from single sample data [11, 53].

Furthermore, to keep the P-SSN graphs from becoming too huge and redundant, we used Cancer Gene Census (CGC, http://cancer.sanger.ac.uk/cosmic/census) to filter out some of the genes before calculating the P-SSNs. The CGC comprises evidence-based, manually-curated summaries of 719 genes and describes genes characterized by somatic or germline mutations in their coding regions in most cases [54]. After significance tests during the construction of P-SSNs, the remaining 710 genes comprised the list of candidate nodes for the P-SSNs. A patient's P-SSN is originally an undirected graph with no weights or node features, and we add the gene expression of this patient to each node as a feature. To ensure that the expressions of different genes are comparable, we normalized the expressions of each gene to ensure that their values are in the range of [0,1]. Table 2 summarizes the P-SSNs of the four TCGA cohorts.

Methodology

Hierarchical graph neural network

Preliminaries

Problem statement Let $G(V, E, X)$ be a graph with $N = |V|$ nodes and $|E|$ edges, where V is the set of nodes, E is the set of edges. Each node $v \in V$ has a d dimensional feature represented by $x \in X$. The node feature matrix of G is denoted by $X \in R^{N\times d}$, and its adjacency matrix is denoted by $A \in R^{N\times N}$. Given a dataset $(G_1, y_1), (G_2, y_2), \cdots, (G_T, y_T)$ of size T, where y_i is the label of Sample i, the task of graph classification is to learn a mapping $\mathcal{F}$ from the graph set $\mathcal{G}$ to the label set $\mathcal{Y}$. When there are several layers in the GNN, the graph can be represented as $G\left(V^l, E^l, H^l\right)$ and A^l of layer l, where H is the matrix of node representation, assuming that $H^0 = X$ and the maximum number of layers is L.

Graph convolution networks We use GraphSAGE [17] to update the representation of the nodes, learn the cluster assignment matrix, and the soft masks. GraphSAGE can be described as follows:

$$h_v^l = \sigma\left(\boldsymbol{W^l} \cdot \mathrm{CONCAT}\left(h_v^{(l-1)}, h_{\mathcal{N}}(v)^l\right)\right), \quad (1)$$

Table 2 A Summary of P-SSNs of Four TCGA cohorts

	BRCA	STAD	LUAD	COAD
# Graphs	1101	352	532	421
# Nodes	517,339	208,678	218,773	210,044
# Edges	13,621,278	5,340,408	3,250,282	2,647,694
Avg. # of nodes per graph	469.88	592.84	411.23	498.92
Avg. # of edges per graph	12,371.73	15,171.61	6109.55	6289.06

$$h^l_{\mathcal{N}(v)} = f^l_{AGG}\left(\left\{h^{l-1}_u, \forall u \in \mathcal{N}(v)\right\}\right), \tag{2}$$

where $\sigma(\cdot)$ is a non-linearity function, W is the weight vector, $f_{AGG}(\cdot)$ is an aggregation function and $\mathcal{N}(\mathrm{v})$ is the node v's neighbors. As the most commonly used aggregation function and the default option of Python libraries for GCNs, the Mean aggregation function is used for GraphSAGE in our model [21].

Hierarchical pooling A GNN learns the mapping between nodes at the current level and nodes at the next level in the structure of a standard differentiable graph pooling module. This node-to-node assignment correlation can be thought of as soft cluster assignment matrix learning for all nodes [21, 23]. Similarly, the embedding representation of the mapped clusters also relies on the input nodes and the cluster assignment matrix. When the connection relationships between clusters are obtained, the coarsened graph of the next layer is formed. This process can be described by (3–5), in which S is the soft cluster assignment matrix for nodes:

$$S^l = softmax\left(GraphSAGE_{l,\,pool} \times \left(A^l,\, H^l\right)\right),\ S^l \in R^{N_l \times N_{l+1}}, \tag{3}$$

$$A^{l+1} = \left(S^l\right)^T A^l S^l,\ A^{l+1} \in R^{N_{l+1} \times N_{l+1}}, \tag{4}$$

$$H^{l+1} = \left(S^l\right)^T GraphSAGE_{l,emd} \times \left(A^l,\, H^l\right),\ H^{l+1} \in R^{N_{l+1} \times d_{l+1}.} \tag{5}$$

Network architecture

The architecture of the HGNN for classification and significant subgraph detection is shown in Fig. 1. The P-SSN approach converts gene expression data into graphs, which are then utilized as input to the hierarchical pooling and graph classification pipeline. We use a soft mask layer after the first pooling module to evaluate a factor for each supernode, which is used to deflate the node representation. There are $L-1$stacked graph pooling modules in the hierarchical pooling phase, where the original input and soft mask-trimmed data form two channels that flow in parallel. Due to the problem of over-smoothing in GCNs, the depth of the network L, i.e., the total number of convolution-pooling modules combined, is set to 3; the last convolutional layer is not followed by the pooling operator and is therefore regarded the $L+1$layer. In addition to concatenating the outputs of each layer at the end and processing them using the last MLP as in other graph pooling frameworks, each convolution-pooling module in our model is followed by a readout module and an additional supervised learner. Finally, the results of the original input and the learned masks are used for subsequent perturbation and verification.

One obvious disadvantage of standard differentiable graph pooling operators is that differentiable pooling produces dense graphs, which increases computational complexity. It is difficult to compute and store these data in a scalable manner for large graphs. In this case, the supernodes are not regular communities of nodes; the nodes covered by the communities are identical, and only the assignment coefficients differ. As a result, the model is unable to identify genes that are more closely related to one another and thus classify such genes into different gene clusters; it is also unable to completely separate a gene cluster to be evaluated from other gene clusters and thus, it cannot evaluate the impact of a gene cluster on classification independently. In addition, several auxiliary objectives, such as link prediction and cluster assignment entropy regularization, are required for training, which are frequently unrelated to the original objectives as well as time-consuming.

To address these issues, we propose a novel differentiable pooling module to sparsity the learned cluster assignment matrix and reduce the size of graphs and subgraphs. After computing the cluster assignment matrix via GNN, we add a truncation function to the differentiable pooling operator. This function restricts each gene cluster to a set number of genes, avoiding redundant gene interrelationships with other gene clusters. Additional supervised classifiers and elaborate loss functions are used to facilitate training. In addition, we propose an attention-based readout module to obtain the graph representation. Figure 2 depicts the architecture of our proposed graph pooling. Two separate graph convolutions are conducted between input and pooling for learning the representation and the cluster assignment matrix, respectively. The truncation function in pooling removes any excess

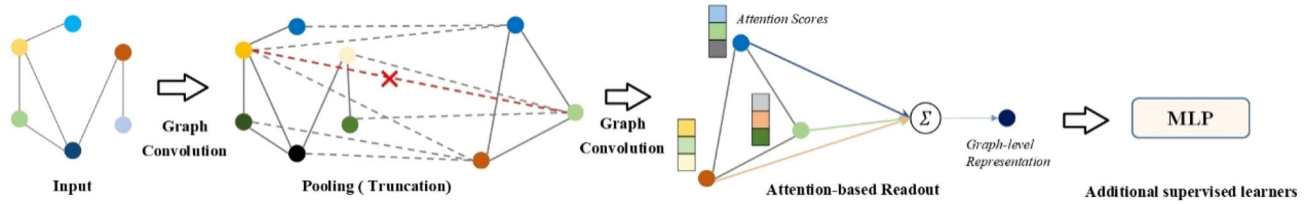

Fig. 2 The architecture of hierarchical pooling

cluster assignments from each node or cluster. To obtain a representation of the graph, the readout module weights and sums the representations of all nodes based on the learned attention scores.

Truncation To reduce the density of the graphs after repeated pooling, we truncate the matrix Sbefore the SoftMax function, only preserving the largest Kassignment coefficients for each node or each cluster and the other values will be truncated to zeros, resulting in a sparse graph structure. In general, a dense cluster assignment matrix will result in coefficients between any node and any cluster, leading to a complete bipartite graph of the set of nodes and the set of clusters. Horizontal truncation allows each node to participate in the representation of up to Kclusters, whereas vertical truncation reserves up to Knodes for each cluster. The information of each node can be assured to converge to some supernodes in the following layer in the horizontal truncation, which generally maintains the highly relevant assignment connections and more information and is more appropriate for classification tasks. Although it is guaranteed that different clusters no longer cover exactly the same nodes, the number of nodes contained in different clusters in the horizontally truncated assignment matrix also varies greatly. Vertical truncation, on the other hand, may manage the size of supernodes so that each supernode has a comparable amount of information and partially overlapping nodes, allowing for more reliable evaluation and making it more ideal for the task of detection key node clusters. To summarize, truncation is a regular sparsification that reserves a fixed number of cluster assignments to ensure that the graph after pooling is not so dense that it cannot be saved or computed for large graphs, while forming several clusters with partial rather than complete overlap.

Additional supervised learners To improve model training and convergence, we add MLPs as extra supervised learners after each readout operation [24]. These MLPs use a three-layer structure that generates classification losses that add up to the overall loss, and their predictions vote to produce the final predictions during testing. Furthermore, with the addition of the mask layer, the loss caused by the masked data input to these classifiers is incorporated into the total loss in (6) and used to guide the learning of a mask that retains more information.

$$loss = CE\left(y_i,\ p_i^{L+1}\right) + \sum_{l=1}^{L}\left(CE\left(y_i,\ p_i^{l}\right) + CE\left(y_i,\ p_{i,mask}^{l}\right)\right), \tag{6}$$

where $CE(\cdot)$ is the cross entropy loss function, $p_i^l$and $p_{i,mask}^l$is the prediction value of sample i for original input and masked input of layer l, and $p_i^{L+1}$is the prediction value of last MLP.

Attention-based readout The readout is a global pooling operation, and the three most commonly used aggregators are Max, Mean, and Sum, which are theoretically ranked in ascending order by expressive power over a multiset [55]. However, when concatenating multiple depths of readout results, which are affected by the number of nodes, the Sum aggregator is more likely to result in higher magnitude disparities, therefore Max and Mean concatenating is an acceptable and practical alternative. We designed a readout module based on an attention mechanism that computes scores for each node in the graph using a GNN and then sums all nodes weighted by the scores to generate a graph-level representation. When calculating several importance scores for a node, the weight is set as the average of the scores, as shown in (7):

$$score_v^l = \frac{1}{R}\sum_{r=1}^{R} GraphSAGE_{l,att}\left(h_v^l\right), \tag{7}$$

where R denotes the dimension of the attention-based score vector, a hyperparameter also called the number of attention heads. This attention mechanism can be regarded as global attention [56, 57]. We also design a variant structure based on the local attention: a virtual node, denoted as g, is added to represent the entire graph G, its representation is the concatenating of Max and Mean readout, and the virtual node is directly connected to every node. The attention coefficients between any one node and the virtual node are then calculated by concatenating the node representation and the virtual node representation, as shown in (8) and (9):

$$score_v^l = \frac{1}{R}\sum_{r=1}^{R} GraphSAGE_{l,att}\left(h_v^l, h_g^l\right), \tag{8}$$

$$h_g^l = CONCAT\left(max\left(H^l\right), mean\left(H^l\right)\right). \tag{9}$$

Mask layer A GNN and an MLP learn a soft mask for each node in the mask layer. A sigmoid function guarantees a value between 0 and 1 for soft masks [33]. We also add a max–min normalization operation after the sigmoid to rearrange the distribution of all masks on [0,1] to differentiate the masks of different nodes. Soft-mask strategies offer a differentiable and interpretable solution to the problem of task-relevant structural information becoming mixed up with irrelevant and noisy parts and becoming indistinguishable for downstream processing. The soft mask strategy, according to the theoretical analysis, could extract any desired substructures or hierarchical structures by learning the graph representation from a sequence of individual subgraphs of the original graph [33]. The mask layer can be used after any combination of embedding and pooling, or it can be applied directly to the input graph. In practice, the mask layer is added after the first pooling layer after the input graph. In this way, the supernodes filtered by soft masks are composed of gene nodes in the input graph rather than supernodes at other levels and can be interpreted as natural gene communities. The soft mask and the node representation are multiplied to deflate the node representation, and the deflated data are referred to as the masked data. The masked graph is treated as another channel of model input and participates in the subsequent computation alongside the original graph. In the case where the masked data produce a different classification than the original data, we add an L1 penalty term to the loss as (10). An important function of soft masks is to identify significant subgraphs and mask unimportant nodes, even those that are against correct classification [58].

$$loss_{mask} = loss + \sum_{l=1}^{L}\left|p_i^l, p_{i,mask}^l\right|, \tag{10}$$

where $|\cdot|$ is the L1 distance.

Subgraph perturbation

The discovery of key gene clusters in cancer staging can be viewed as the interpretability of GNNs in this work, with the goal of finding and confirming which genes play important roles in classifying early- and late-stage cancers. Because it is hard to exhaust and evaluate all subgraphs, we conduct the model explanation and significant subgraph extraction as the following steps in our work: 1) create a graph neural network with a mask layer and hierarchical pooling layers which are described in the previous subsection and train the network end-to-end to extract the significance subgraphs for each instance; 2) reconstruct the datasets, deduct the subgraphs to generate perturbations, and assess the significance of the subgraphs by the impact of perturbations on the classification; 3) group the instances into classes and observe how the significance subgraphs are enriched and explore the patterns or motifs that may be embedded in the subgraphs using powerful bioinformatic tools. In this procedure, HGNN predicts the classification of the input graph, the genes are organized into subgraphs by the first pooling module, and the soft mask layer measures the extent to which different subgraphs contribute to the subsequent computational process. Finally, perturbation demonstrates that removing screened subgraphs reduces confidence of the in properly classifying samples, implying that the screened subgraphs, i.e. key gene clusters, are more essential in determining cancer stage classification.

Perturbation

We sort the gene clusters based on the value of the mask and choose a selection of the top-ranking gene clusters as key gene clusters. We subtract the key gene clusters of each P-SSN from the input graphs, reassemble the dataset, and test the classification using the same trained model. This process is similar to the experiments for quantitative evaluation of graph neural network interpretation methods [29, 31, 32, 59]. In these quantitative evaluations, one of the most common evaluation metrics is fidelity, which is computed to reflect the assumption that the exclusion of salient features identified through explanations should decrease classification accuracy [29, 31, 59]. More precisely, fidelity is defined as the difference in accuracy obtained by masking the significant nodes identified by the model. However, fidelity is a measure of the impact of interpretation on classification in terms of a category or global perspective and cannot be used to quantify the impact of a significant subgraph in a single sample on that sample's classification. Two critical metrics in classification for a single sample are the two dimensions of the vector of the neural network's last MLP output, and they serve as the foundation for the model to evaluate whether this sample is positive or negative. We refer to the two numbers as positive and negative classification values since they do not exactly fit the definition of probability. Inspired by the fidelity, we quantify the effect of removing significant subgraphs on the classification of individual samples by calculating the change in classification values. In contrast to accuracy, the classification values of different samples are not comparable and might vary greatly in value. To make the differences in classification values comparable across samples, we define this metric as the ratio of the difference in classification values to its original value. To quantify the effect of perturbations on classification, we define the percentage change in positive and negative classification values, i.e. the

ratio of the difference between the classification values of perturbed data and the classification values of original data to the absolute value of the original classification values. For two samples with positive classification values of 0.2 and 0.02, we can determine that if the positive classification values of both samples are reduced by 0.1 and 0.01 respectively, the percentage changes of both samples are at a comparable scale, around 50%, despite the tenfold difference in numerical values. The output of the MLP in the last layer for the original data is denoted as $(clsv_{pos}^{L+1}, clsv_{neg}^{L+1})$, the output for perturbed data is denoted as $\left(clsv_{per,pos}^{L+1}, clsv_{per,neg}^{L+1}\right)$, and the percentage change (PC) in positive classification value is defined as (11)

$$PC_{pos} = \frac{clsv_{per,pos}^{L+1} - clsv_{pos}^{L+1}}{\left|clsv_{pos}^{L+1}\right|}, \tag{11}$$

where $|\cdot|$ signifies the absolute value function and the percentage change in negative classification value (PC_{neg}) has the same definition. After deleting the significant subgraphs from positive samples, the model's confidence in classifying the sample as positive should diminish and the negative classification value may rise; the converse is true for negative samples.

Enrichment

To balance the generality of interpretability across individual samples and class collection [60], we group samples into classes to obtain sets of genes that have some commonality in each class of data. We outline a certain range of significant subgraphs for each sample, then calculate the frequency of genes in all/positive/negative samples respectively, and choose genes according to frequency. The Database for Annotation, Visualization and Integrated Discovery (DAVID, https://david.ncifcrf.gov/) [61, 62], is then employed to annotate for Gene Ontology (GO) and examine these genes for biological process (BP), molecular function (MF), cellular component (CC) enrichment, and KEGG (Kyoto Encyclopedia of Genes and Genomes) pathway enrichment.

Results and discussion

The classification accuracy in cancer staging

We outline the baselines, several variants of our model, and the experiment settings in this section. Classification model performance is evaluated using accuracy and training time. The best results are reported in Table 3, where each result is the best outcome following several parameter tunings.

Table 3 Best accuracy for all datasets

Model	BRCA	STAD	LUAD	COAD
GraphConv	0.8420	0.7198	0.6390	0.8873
GIN	0.8665	0.8008	0.6461	0.9022
SAGpool	0.8329	0.6047	0.6461	0.8549
ASAP	0.8202	0.5801	0.6318	0.8289
DGCNN	0.8111	0.5344	0.6342	0.8363
Graph U Net	0.8511	0.5680	0.6248	0.8739
GMN	0.8647	0.5428	0.6393	0.8834
Diffpool	0.8856	0.7937	0.6604	0.9267
Ours (*global_att*)	**0.9028**	0.8402	**0.6891**	**0.9436**
Ours (local_att)	0.9010	**0.8575**	0.6822	0.9379
Ours (soft mask)	0.8955	0.7551	0.6508	0.9080

The best results on each dataset are indicated in bold

Baselines

We selected a list of recent GNNs and graph pooling methods as baselines, including: GraphConv [63], GIN [55], DGCNN [25], Graph U-Net [26], SAGpool [27], ASAP [22], GMN [64], and Diffpool [21]. GraphConv and GIN are mix of GCNs and readout modules, while the others are GNNs with pooling modules. These models were trained according to the architecture and settings suggested in the literature, using code provided by the authors or PyTorch Geometric [65].

Variants

By default, the classification model without a mask layer employs horizontal truncation and global attention readout layers (*global_att*), while a variant for classification employs horizontal truncation and local attention readout layers (*local_att*). A network with a mask layer and vertical truncation in each pooling layer (*soft mask*) is the option for the discovery of key gene clusters.

Experimental settings

The tenfold cross-validation was used for all datasets. The maximum number of epochs of training is 1000, and we use an early stop technique (threshold set to 100). One-ninth of the data from the training set is taken as the validation set for the early stop. Using the Adam optimization approach, the starting learning rate is 0.001. The batch size for training is tuned in {8, 16, 32, 64}, and the size of GCN layers is also searched in {8, 16, 32, 64}. The most essential parameter in all pooling operators is the pooling ratio, which affects how many supernodes are generated or how many nodes are kept after pooling. On the one hand, a lower pooling ratio might accelerate computation and fast reduce graph size, but it may

eliminate a huge amount of information, reducing classification performance. A higher pooling ratio, on the other hand, may produce networks with complicated structures and preserve more node information, but the cost of computation and storage is higher. The pooling ratio is set to 0.4 for all models for a trade-off between performance and computational requirement [22, 28]. All models were trained using one NVIDIA GeForce RTX 3060 GPU.

Table 3 shows the experimental results. DiffPool outperforms all baselines, with GIN coming in a close second. The former is a standard hierarchical pooling method for learnable clustering strategies, whereas the latter is simply a GCN-readout combination. On these four TCGA cohorts, the average accuracy of the baselines was 0.8468 for BRCA, 0.6430 for STAD, 0.6402 for LUAD, and 0.8742 for COAD. On the BRCA and STAD datasets, the DGCNN performs the worst, and it also does poorly on other datasets. In part, this is because its strategy, which only selects a small number of nodes and does not preserve the features of eliminated nodes, does not learn the classification patterns well. Another notable fact is that, while pooling is critical for the graph classification task and the pooling method DiffPool gets the best results in the baselines, not all pooling methods always beat the GCN-readout combination approaches in classification. In other words, the introduction of graph pooling does not always directly improve classification [66]. Graph pooling, on the other hand, remains to have considerable advantages and attractiveness in multi-task algorithms such as node community partitioning and representation learning.

In addition, we have collected some relevant results reported in the literature. ECMarker, a machine learning approach, achieves a classification accuracy of 0.48 for a classification task involving four phenotypes, including early-/late- stage in LUAD and LUSC (Lung Squamous Cell Carcinoma), compared to a baseline of 0.27 [67]. Roy et al. focused on Invasive Ductal Carcinoma (IDC), a specific subtype of breast cancer, and experimented with 610 patients from the BRCA cohort, screening candidate genes and utilizing various machine learning models for early/late stage prediction, achieving the highest accuracy of 92% [68]. Similarly, Ma et al. considered four TCGA cohorts for early/late stage cancer classification, evaluating several machine learning approaches such as extreme gradient boosting (XGBoost), support vector machine (SVM), random forest (RF), and others [69]. XGBoost obtained the highest classification accuracies of 0.752 for KIRC (Kidney Renal Clear Cell Carcinoma), 0.875 for KIRP (Kidney Renal Papillary Cell Carcinoma), 0.602 for HNSC (Head and Neck Squamous Cell Carcinoma), and 0.478 for LUSC. Among these studies, traditional machine learning methods are the most prevalent, whereas deep learning, particularly graph neural networks, is infrequently exploited. And, as a result of a lack of data volume, the curse of dimensionality, and data imbalance, these approaches do not always yield particularly satisfying results and typically consist of a two-stage analysis process: screening for genes and classification for stages.

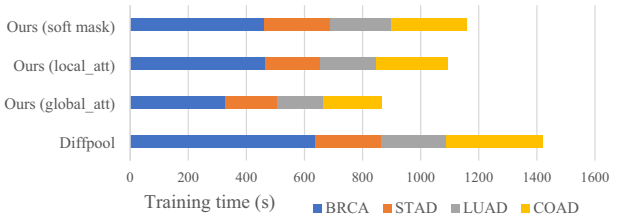

Fig. 3 Average training time for models with differentiable pooling operators

Experimental results show that our classification model rivals advanced models in classification abilities and outperforms them on several datasets, reaching over 90% in BRCA and 94% in COAD. The variation used to extract subgraphs has somewhat worse classification performance than the model used just for classification, but it still surpasses the majority of baseline models. This fluctuation in the classification accuracy exemplifies soft masks' ability to keep the great majority of information and so have some but not excessive impact on representation aggregation. In addition to better classification performance, HGNN has a lower computational cost when compared to the standard differentiable pooling operator. The average training time for models with differentiable pooling operators on the four datasets is shown in Fig. 3. When compared to the standard differentiable pooling operator with an average total training time of 1420.71 s, both the HGNN for classification and the HGNN with mask have shorter training time, demonstrating the feasibility of our method in terms of computational resources.

Most models perform poorly on STAD, whereas our model and Diffpool perform better. This may be attributed to the fact that the STAD dataset has fewer samples with more nodes and edges, and the hierarchical pooling structure can learn different levels of structure well without missing important information. A similar issue exists with the COAD dataset, and the imbalance in the COAD data is more severe than in STAD, resulting in poorer model performance on this dataset. Another important factor influencing classification effectiveness is tumor heterogeneity: not only across individuals but also within each tumor, as evidenced by the fact that cancers can be split into different subtypes based on the mechanism at the pathophysiology and molecular level [70]. The model of Roy et al. performed better in part because it focused on only one subtype and used the Synthetic Minority Oversampling Technique (SMOTE) approach to handle data imbalance and lessen the detrimental effects of heterogeneity and data imbalance [68].

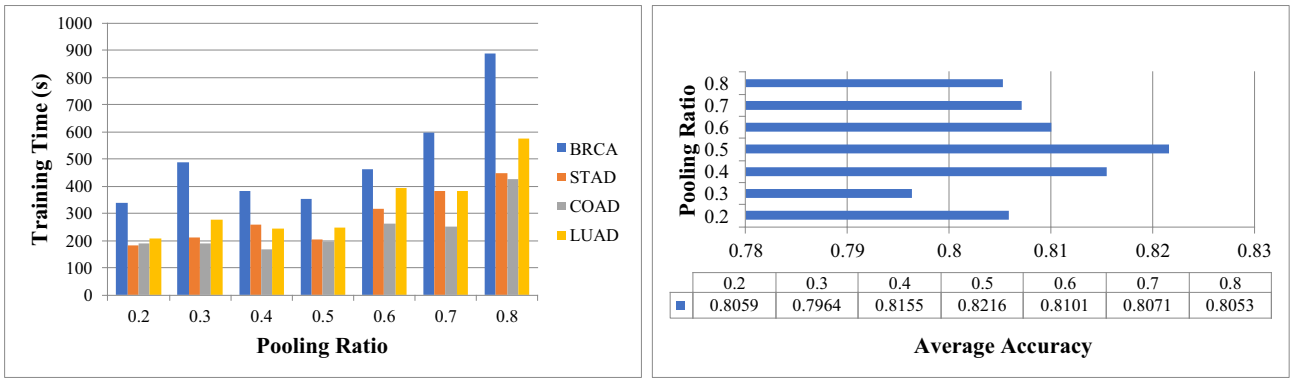

Fig. 4 Training time and average accuracy of different pooling ratios

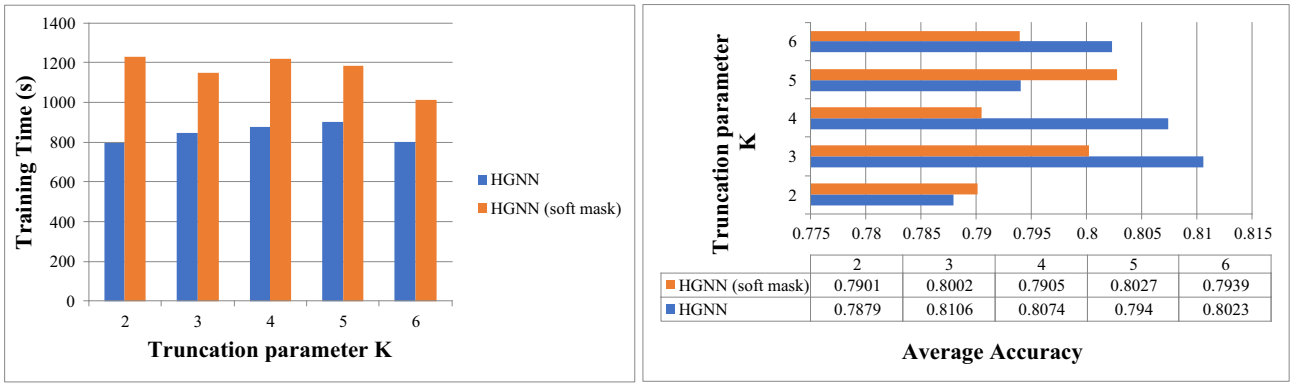

Fig. 5 Training time and average accuracy of different values of K

Parameter analysis

We performed a grid search for two hyper-parameters, batch size and GCN layer size, and employed the average accuracy among all four datasets as an evaluation metric. The results suggested that 32 is the best option for both batch size and GCN layer size. Fixing other parameters, we tested the sensitivity for two crucial hyperparameters, the pooling ratio and the number of genes retained in the truncation function, K. The model's training time is used to measure the model's computational efficiency and the computational resources consumed. The pooling ratio is tuned in {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}, and Fig. 4 depicts the training time and accuracy of the classification model with varying pooling ratios. It can be shown that when the pooling ratio increases, the training time increases significantly across all datasets, and the classification accuracy increases and subsequently decreases, with a maximum at pooling ratios of 0.4 and 0.5. Given that the training time is also locally minimum with a pooling ratio of 0.4 or 0.5, these two values are optimal for the pooling ratio and consistent with prior literature reports. The parameter K in the truncation function is tuned in {2,3,4,5,6}, and Fig. 5 depicts the classification model and variant with a mask layer's training time and accuracy. Because of its more complex model architecture and more parameters, HGNN with a soft mask layer obviously takes longer. In addition, the HGNN with mask layer achieves the highest classification accuracy at $K = 5$ and will thus be employed for subsequent analysis.

Discovery of key gene clusters

We calculated the average percentage change in positive and negative classification values across the positive and negative sample sets, with positive numbers representing an increase and negative numbers representing a decrease, and plotted them in Fig. 6. The vertical axis represents the ratio of the difference between the perturbed classification value and the original classification value to the absolute value of the original classification value. The horizontal axis represents the number of significant subplots removed. The truncation parameter K in the model used to discover key gene clusters is set to 5, implying that each subgraph contains up to 5 genes. When K becomes too small, or even approaches 1, the model degenerates into a method for selecting node strategy, and the capacity to discover node communities is lost. When K is too large, interference occurs when conducting experiments to reconstruct the data set, and it is possible that

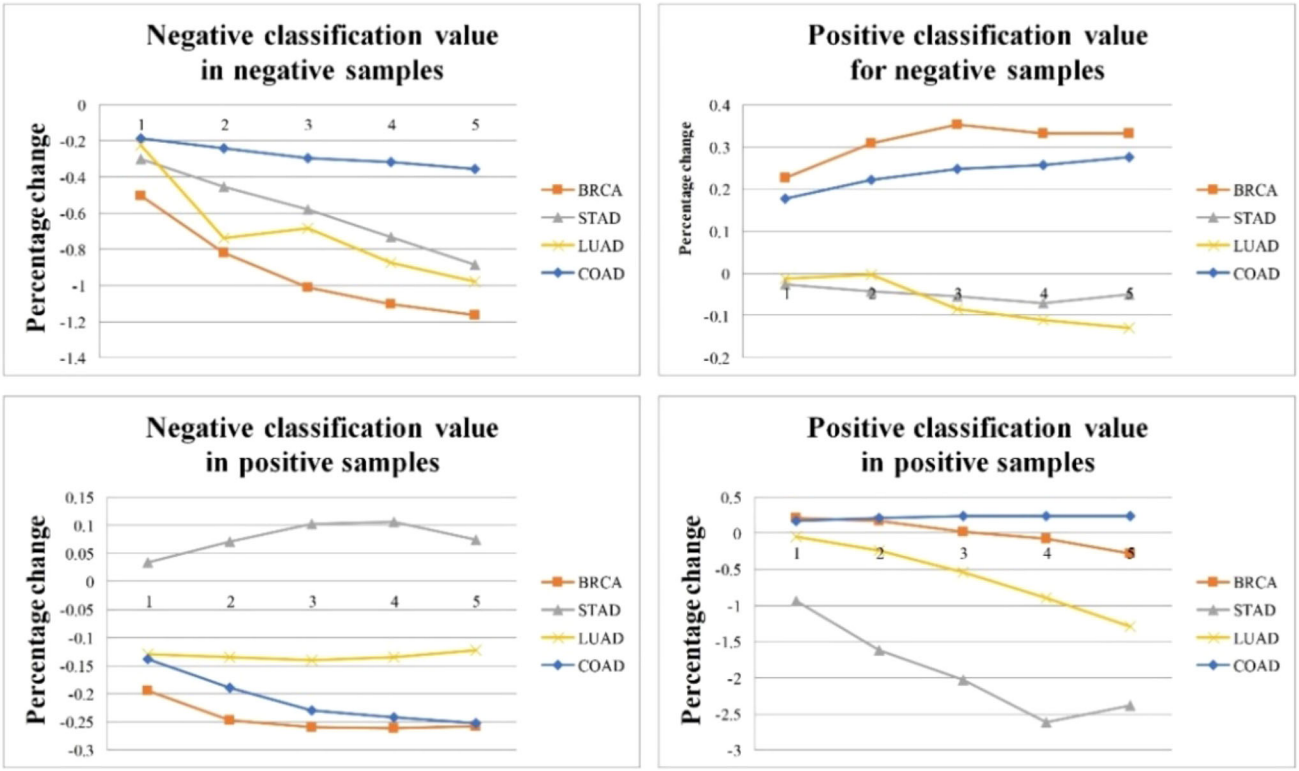

Fig. 6 Average percentage change in positive and negative classification values for the positive and negative samples

removing too many nodes rather than the key ones affects the model's judgment of the data.

The majority of the classes meet the expectation that taking out important elements reduces the model's confidence in the correct classification and thus affects the classification results. The percentage changes in negative classification values in negative samples and positive classification values in positive samples are mostly less than 0. This shows that the model's confidence in classifying negative samples as negative and positive samples as positive is dwindling. The absolute value of the percentage change rises as the number of removed significant gene clusters rises, indicating that the model's confidence in correct classification decreases progressively. And as the number of removed subgraphs increases, the structure of the graph changes so dramatically that the model has difficulties in recognizing it, and thus cannot determine whether it is unrecognizable due to the significant subgraphs. The imbalance of classes in datasets, on the other hand, may cause the model to learn patterns that are biased toward one class. It should also be noted that the classification values of misclassifications just have the opposite trend in some datasets, such as negative samples in BRCA and COAD, and positive samples in STAD. The percentage changes in the other datasets are also negative and generally steady, not fluctuating with the number of deleted nodes. However, there is no universal rule that always applies to the percentage change of positive classification values in negative samples and negative classification values in positive samples. On the one hand, this is because positive and negative classification values are not intrinsically constant in total value, and there is no connection between them for trading off and taking turns. Removing key structures for accurate classification, on the other hand, causes the model to fail to recognize the proper classification of a sample rather than entices the model to misidentify it.

The model's classification of the data may alter when the most significant subgraphs are eliminated, which is a specific situation. We specifically looked for instances where the classification changed from correct to incorrect, and we discovered a total of 56 instances. A negative sample from STAD is an intriguing example because the reduction in the negative classification value is 315,417 times greater classification value itself. This resulted in the sample being classed as positive by the classifier, whereas the classifier had previously properly identified it; the percentage drop in the negative classification value of this sample is treated as an outlier which is not truly shown in Fig. 6. Its significant subgraph includes the genes SKI, XPO1, DDX5, KDSR, and DEK, three of which are involved in the biological process of negative regulation of transcription from RNA polymerase II promoter. These genes show great individual specificity, occurring only five

times in total in the most significant subgraphs of the other samples, and XPO1 is even unique to this sample.

We also counted the frequency of the top 5 significant subgraphs' genes in each class of samples and noticed the enrichment of significant genes in each class. The three genes with the highest frequencies are RHOA (62.59%, late-stage, BRCA), CTNNA2 (53.64%, late-stage, LUAD), and SIX2 (52.73%, late-stage, LUAD). RHOA is a member of the Ras superfamily of small GTPases, and it is one of three RHO proteins (A, B, and C) that play important roles in regulating cytoskeletal organization, directional migration, and tumor cell motility [71]. RHOA overexpression is commonly described in breast cancer, and strong RHOA expression and low RHOB expression are associated with the basal-like subtype of breast cancer [71, 72]. It has also been shown that reduced RHOA expression enhances breast cancer metastasis with a concomitant increase in CCR5 and CXCR4 chemokines signaling [73]. According to reports, the CTNNA2 mutation is involved in the adhesion junction pathway and has previously been identified as a tumor suppressor in laryngeal cancer, and its inactivation in head and neck squamous cells (HNSC) is related to migration and invasion advantages [74, 75]. CTNNA2 was identified as a new mutated gene in LUAD, related with prolonged overall survival in LUAD patients, and can be connected with tumor growth, maintenance, and progression in several experiments based on somatic mutation data [74, 76]. In addition, SIX2 expression was significantly increased in non-small cell lung cancer (NSCLC) tissues and Kaplan–Meier plotter analysis showed that six2 expression was negatively correlated with the survival of lung cancer patients [77]. And SIX2 expression was shown to be highly correlated with the TNM stage of NSCLC, with higher expression in advanced tumor stages [78]. It was discovered that SIX2 suppressed caspase-8 mediated cell death as a potential mechanistic explanation for cancer cell resistance of NSCLC, whereas SIX2 knockdown enhanced cisplatin sensitivity in parental NSCLC cells and attenuated cisplatin resistance in cisplatin-resistant NSCLC cells [77, 79]. In conclusion, there is considerable literature supporting the association of genes RHOA, CTNNA2, and SIX2 with the development and progress of cancer, particularly in the late stages of cancer, which is consistent with our findings [73, 74, 78].

Table 4 summarizes some processes with stage specificity after querying the KEGG pathway, BP, MF, and CC enrichment of the 20 most frequent genes. Stage-specificity has two meanings: first, each enrichment result appears in only one type of cancer, and results that appear in multiple cancers are removed; second, each result appears in only one specific stage of the same cancer, and results that appear in both early and late stages are also removed. The potential relationship between some of these enrichment results and the corresponding cancers has attracted the attention of researchers. For example, the KEGG pathways hsa04722 "Neurotrophin signaling pathway" and hsa04530 "Tight junction" are found to be enriched in the late stages of BRCA, while other studies have found that neurotrophin expression and regulation contribute to chemotherapeutic resistance in breast cancer cells, and tight junctions may play an important role in the intermediate link of metastasis in breast cancer [80, 81]. The enrichment of GO:0,002,467 "germinal center (GC) formation" in the early-stage of LUAD is consistent with a survey of immune cell infiltration phenomenon in NSCLC patients' histologic sections: patients with stage I NSCLC had a higher prevalence of intratumoral GCs than patients with other stages, and intratumoral GCs are associated with early-stage NSCLC [82]. ACKR3 (CXCR7) and CCR7 are common genes involved in GO:0,016,493 "C–C chemokine receptor activity" and GO:0,019,957 "C–C chemokine binding", and as chemokine receptor families, they are both thought to play an important role in colorectal carcinoma metastasis and invasion [83, 84]. Furthermore, CXCR7 promotes colon cancer growth by targeting the vascular endothelial growth factor via the AKT/ERK pathway to regulate angiogenesis in colon cancer [85]. Enrichment of these two genes and products in the early stages of COAD may indicate cancer development as well as metastasis to more malignant cancers. Overall, many of the enrichment results we observed are consistent with previous studies and observations, and some of the new findings we present may provide new insights as leads for future biological exploration.

Conclusions

Early cancer screening and individual-specific discovery of contributing gene sets play a significant role in cancer personalized medicine. This study presents a computational flow and deep learning model for cancer stage classification and extraction of important gene clusters end to end. Experiments are carried out on four genuine datasets from TCGA to acquire reliable prediction interpretation and identify relevant biological insights by perturbing the datasets.

One limitation of our method for real-world applications is the scarcity of available samples, although the method has achieved state-of-the-art performance on small datasets. The model's efficacy could be improved when working with larger datasets. Additionally, the key genes identified by the model are significant genes that distinguish the gene networks of cancer patients at different stages, equivalent to network biomarkers. Further research into the biological roles of candidate genes can facilitate the understanding of how these genes drive cancer development, thus allowing the model's findings to be translated into credible CDGs.

Table 4 Stage-specific KEGG pathway and GO, MP, and CC enrichment ($P \leq 0.05$)

Cancer	Stage	KEGG pathway (*P* value)	BP (*P* value)	MF (*P* value)	CC (*P* value)
BRCA	Early	hsa01521: EGFR tyrosine kinase inhibitor resistance (0.0067)	GO:0,038,133 ~ ERBB2-ERBB3 signaling pathway (0.0037) GO:0,038,135 ~ ERBB2-ERBB4 signaling pathway (0.0047) GO:0,060,397 ~ JAK-STAT cascade involved in growth hormone signaling pathway (0.0093)	GO:0,043,125 ~ ErbB-3 class receptor binding (0.0050) GO:0,000,405 ~ bubble DNA binding (0.0081) GO:0,051,015 ~ actin filament binding (0.0213)	GO:0,048,471 ~ perinuclear region of cytoplasm (0.0277)
	Late	hsa04928: Parathyroid hormone synthesis, secretion and action (0.0009) hsa04722: Neurotrophin signalling pathway (0.0012) hsa04530: Tight junction (0.0033)	GO:0,030,036 ~ actin cytoskeleton organization (0.0007) GO:0,051,496 ~ positive regulation of stress fiber assembly (0.0013) GO:0,034,446 ~ substrate adhesion-dependent cell spreading (0.0015)	GO:0,051,022 ~ Rho GDP-dissociation inhibitor binding (0.0038) GO:0,004,672 ~ protein kinase activity (0.0053) GO:0,016,301 ~ kinase activity (0.0193)	GO:0,098,794 ~ postsynapse (0.0052) GO:0,005,938 ~ cell cortex (0.0100) GO:0,070,062 ~ extracellular exosome (0.0125)
STAD	Early	hsa05235: PD-L1 expression and PD-1 checkpoint pathway in cancer (0.0128) hsa04550: Signaling pathways regulating pluripotency of stem cells (0.0312)	GO:0,048,679 ~ regulation of axon Regeneration (0.0049) GO:0,030,879 ~ mammary gland development (0.0215) GO:0,071,425 ~ hematopoietic stem cell proliferation (0.0234)		

Table 4 (continued)

Cancer	Stage	KEGG pathway (*P* value)	BP (*P* value)	MF (*P* value)	CC (*P* value)
	Late		GO:0,048,671 ~ negative regulation of collateral sprouting (0.0098) GO:1,903,800 ~ positive regulation of production of miRNAs involved in gene silencing by miRNA (0.0117) GO:0,001,502 ~ cartilage condensation (0.0215)	GO:0,045,499 ~ chemorepellent activity (0.0289)	
COAD	Early		GO:0,002,357 ~ defense response to tumor cell (0.0127) GO:0,001,913 ~ Tcell-mediated cytotoxicity (0.0127) GO:0,008,284 ~ positive regulation of cell proliferation (0.0159)	GO:1,990,837 ~ sequence-specific double-stranded DNA binding (0.0175) GO:0,016,493 ~ C–C chemokine receptor activity (0.0229) GO:0,019,957 ~ C–C chemokine binding (0.0240)	GO:0,005,667 ~ transcription factor complex (0.0191) GO:0,009,986 ~ cell surface (0.0201)
	Late		GO:0,046,321 ~ positive regulation of fatty acid oxidation (0.0059) GO:0,007,167 ~ enzyme-linked receptor protein signaling pathway (0.0069) GO:0,002,250 ~ adaptive immune response (0.0094)	GO:0,042,393 ~ histone binding (0.0152) GO:0,004,888 ~ transmembrane signaling receptor activity (0.0153) GO:0,001,227 ~ transcriptional repressor activity, RNA polymerase II transcription regulatory region sequence-specific binding (0.0433)	GO:0,019,815 ~ B cell receptor complex (0.0065)
LUAD	Early	hsa05169: Epstein-Barr virus infection (0.0342)	GO:0,002,268 ~ follicular dendritic cell differentiation (0.0020) GO:0,002,467 ~ germinal center formation (0.0098) GO:0,030,198 ~ extracellular matrix organization (0.0113)		GO:0,033,257 ~ Bcl3/NF-kappaB2 complex (0.0017) GO:0,032,991 ~ macromolecular complex (0.0029) GO:0,032,993 ~ protein-DNA complex (0.0397)
	Late	hsa04933: AGE-RAGE signaling pathway in diabetic complications (0.0141)	GO:0,001,822 ~ kidney development (0.0060) GO:0,001,501 ~ skeletal system development (0.0078) GO:0,072,075 ~ metanephric mesenchyme development (0.0079)	GO:0,048,407 ~ platelet-derived growth factor binding (0.0111) GO:0,070,742 ~ C2H2 zinc finger domain binding (0.0150)	

A potential expansion of our study would be to use a more detailed stage division to identify potential biomarkers specific to each stage [86]. In addition, the fixed size and number of learned subgraphs in our model are limited by hyperparameters. Another possibility for future work is to combine our framework with community detection based on GNNs to achieve a more flexible way of subgraph segmentation and assessment [87]. And more intriguing reference gene sets for constructing the network would also stimulate fresh and diverse insights. WebTWAS, for example, is a resource for candidate disease susceptibility genes identified by transcriptome-wide association study (TWAS), and it contains many candidate genes obtained through computational and statistical methods rather than biological or medical experimental observations [88].

Acknowledgements This work was supported by the National Natural Science Foundation of China (No. 62072212), the Development Project of Jilin Province of China (Nos. 20220508125RC, 20210508060RQ), National Key R & D Program (No. 2018YFC2001302), and the Jilin Provincial Key Laboratory of Big Data Intelligent Cognition (No. 20210504003GH).

Data availability Data openly available in a public repository. The data that support the findings of this study are openly available in The Cancer Genome Atlas (TCGA) at https://portal.gdc.cancer.gov.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Sung H, Ferlay J, Siegel RL et al (2021) Global Cancer Statistics 2020: GLOBOCAN estimates of incidence and Mortality Worldwide for 36 Cancers in 185 Countries. CA Cancer J Clin 71:209–249. https://doi.org/10.3322/caac.21660
2. Skrede OJ, de Raedt S, Kleppe A et al (2020) Deep learning for prediction of colorectal cancer outcome: a discovery and validation study. LANCET 395:350–360
3. Taube JM, Galon J, Sholl LM et al (2018) Implications of the tumor immune microenvironment for staging and therapeutics. Modern Pathol 31:214–234. https://doi.org/10.1038/modpathol.2017.156
4. Supplitt S, Karpinski P, Sasiadek M, Laczmanska I (2021) Current achievements and applications of transcriptomics in personalized cancer medicine. Int J Mol Sci 22:1–22
5. Herbst RS, Morgensztern D, Boshoff C (2018) The biology and management of non-small cell lung cancer. Nature 553:446–454. https://doi.org/10.1038/nature25183
6. Hu Z, Ott PA, Wu CJ (2018) Towards personalized, tumour-specific, therapeutic vaccines for cancer. Nat Rev Immunol 18:168–182. https://doi.org/10.1038/nri.2017.131
7. Xu Z, Li X, Stojanovic V (2021) Exponential stability of nonlinear state-dependent delayed impulsive systems with applications. Nonlinear Anal Hybrid Syst 42:101088. https://doi.org/10.1016/j.nahs.2021.101088
8. Liu X, Wang Y, Ji H et al (2016) Personalized characterization of diseases using sample-specific networks. Nucl Acids Res 44:e164. https://doi.org/10.1093/nar/gkw772
9. Jahagirdar S, Saccenti E (2021) Evaluation of single sample network inference methods for metabolomics-based systems medicine. J Proteome Res 20:932–949. https://doi.org/10.1021/acs.jproteome.0c00696
10. Guo WF, Zhang SW, Zeng T et al (2020) Network control principles for identifying personalized driver genes in cancer. Brief Bioinform 21:1641–1662. https://doi.org/10.1093/bib/bbz089
11. Huang Y, Chang X, Zhang Y et al (2021) Disease characterization using a partial correlation-based sample-specific network. Brief Bioinform 22:bbaa062. https://doi.org/10.1093/bib/bbaa062
12. Li Y, Huang C, Ding L et al (2019) Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. Methods 166:4–21. https://doi.org/10.1016/j.ymeth.2019.04.008
13. Wu Z, Pan S, Chen F et al (2021) A Comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32:4–24. https://doi.org/10.1109/TNNLS.2020.2978386
14. Xin X, Tu Y, Stojanovic V et al (2022) Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems. Appl Math Comput 412:126537. https://doi.org/10.1016/j.amc.2021.126537
15. Wei T, Li X, Stojanovic V (2021) Input-to-state stability of impulsive reaction–diffusion neural networks with infinite distributed delays. Nonlinear Dyn 103:1733–1755. https://doi.org/10.1007/s11071-021-06208-6
16. Chen Z, Zhang B, Stojanovic V et al (2020) Event-based fuzzy control for T-S fuzzy networked systems with various data missing. Neurocomputing 417:322–332. https://doi.org/10.1016/j.neucom.2020.08.063
17. Hamilton WL, Ying R, Leskovec J (2017) Inductive Representation Learning on Large Graphs. In: 31st Annual Conference on Neural Information Processing Systems (NIPS). Long Beach, CA
18. Veličković P, Cucurull G, Casanova A, et al (2018) Graph Attention Networks. In: International Conference on Learning Representations
19. Zhou Y, Zheng H, Huang X et al (2022) Graph neural networks: taxonomy, advances, and trends. ACM Trans Intell Syst Technol 13:1–54. https://doi.org/10.1145/3495161
20. Ma Y, Wang S, Aggarwal CC, Tang J (2019) Graph convolutional networks with EigenPooling. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, NY, USA, pp 723–731
21. Ying Z, You J, Morris C, et al (2018) Hierarchical Graph Representation Learning with Differentiable Pooling. In: Bengio S, Wallach H, Larochelle H, et al (eds) Advances in Neural Information Processing Systems. Curran Associates, Inc.
22. Ranjan E, Sanyal S, Talukdar P (2020) ASAP: adaptive structure aware pooling for learning hierarchical graph representations. Proceed AAAI Con Artif Intell 34:5470–5477. https://doi.org/10.1609/aaai.v34i04.5997

23. Yuan H, Ji S (2020) StructPool: Structured Graph Pooling via Conditional Random Fields. In: International Conference on Learning Representations
24. Huang J, Li Z, Li N, et al (2019) Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In: Proceedings of the IEEE International Conference on Computer Vision. Institute of Electrical and Electronics Engineers Inc., pp 6479–6488
25. Zhang M, Cui Z, Neumann M, Chen Y (2018) An end-to-end deep learning architecture for graph classification. Proceed AAAI Conf Artif Intell. https://doi.org/10.1609/aaai.v32i1.11782
26. Gao H, Ji S (2019) Graph U-Nets. In: Proceedings of the 36th International Conference on Machine Learning. pp 2083--2092
27. Lee J, Lee I, Kang J (2019) Self-Attention Graph Pooling. In: Proceedings of the 36th International Conference on Machine Learning. pp 3734–3743
28. Zhang L, Wang X, Li H, et al (2020) Structure-Feature based Graph Self-adaptive Pooling. In: Proceedings of the of The Web Conference 2020. ACM, NY, USA, pp 3098–3104
29. Wang X, Wu Y, Zhang A et al (2021) Towards multi-grained explainability for graph neural networks. In: Beygelzimer A, Dauphin Y et al (eds) Ranzato M. Systems. Curran Associates Inc, Advances in Neural Information Processing, pp 18446–18458
30. Li X, Dvornek NC, Zhou Y et al (2019) Graph Neural Network for Interpreting Task-fMRI Biomarkers. In: Shen D, Liu T, Peters TM et al (eds) Medical image computing and computer assisted intervention – MICCAI 2019. Springer International Publishing, Cham, pp 485–493
31. Pope PE, Kolouri S, Rostami M, et al (2019) Explainability Methods for Graph Convolutional Neural Networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp 10764–10773
32. Wang X, Wu Y, Zhang A et al (2022) Reinforced causal explainer for graph neural networks. IEEE Trans Pattern Anal Mach Intell PP. https://doi.org/10.1109/TPAMI.2022.3170302
33. Yang M, Shen Y, Qi H, Yin B (2021) Soft-mask: Adaptive Substructure Extractions for Graph Neural Networks. In: Proceedings of the Web Conference 2021. ACM, New York, NY, USA, pp 2058–2068
34. Pham VVH, Liu L, Bracken C et al (2021) Computational methods for cancer driver discovery: A survey. Theranostics 11:5553–5568
35. Wang T, Ruan S, Zhao X et al (2021) OncoVar: An integrated database and analysis platform for oncogenic driver variants in cancers. Nucleic Acids Res 49:D1289–D1301. https://doi.org/10.1093/nar/gkaa1033
36. Kan Y, Jiang L, Guo Y et al (2022) Two-stage-vote ensemble framework based on integration of mutation data and gene interaction network for uncovering driver genes. Brief Bioinform 23:bbab429. https://doi.org/10.1093/bib/bbab429
37. Nulsen J, Misetic H, Yau C, Ciccarelli FD (2021) Pan-cancer detection of driver genes at the single-patient resolution. Genome Med 13:12. https://doi.org/10.1186/s13073-021-00830-0
38. Kan Y, Jiang L, Tang J et al (2021) A systematic view of computational methods for identifying driver genes based on somatic mutation data. Brief Funct Genom 20:333–343
39. Yang H, Wei Q, Zhong X et al (2017) Cancer driver gene discovery through an integrative genomics approach in a non-parametric Bayesian framework. Bioinformatics 33:483–490. https://doi.org/10.1093/bioinformatics/btw662
40. Zhang J, Zhang S (2017) Discovery of cancer common and specific driver gene sets. Nucl Acids Res 45:e86. https://doi.org/10.1093/nar/gkx089
41. Zhang T, Zhang SW, Li Y (2021) Identifying driver genes for individual patients through inductive matrix completion. Bioinformatics 37:4477–4484. https://doi.org/10.1093/bioinformatics/btab477
42. Colaprico A, Olsen C, Bailey MH et al (2020) Interpreting pathways to discover cancer driver genes with Moonlight. Nat Commun 11:69. https://doi.org/10.1038/s41467-019-13803-0
43. Chen Z, Lu Y, Cao B et al (2022) Driver gene detection through Bayesian network integration of mutation and expression profiles. Bioinformatics 38:2781–2790. https://doi.org/10.1093/bioinformatics/btac203
44. Sudhakar M, Rengaswamy R, Raman K (2022) Novel ratio-metric features enable the identification of new driver genes across cancer types. Sci Rep 12:5. https://doi.org/10.1038/s41598-021-04015-y
45. Wang J, Yang Z, Domeniconi C et al (2021) Cooperative driver pathway discovery via fusion of multi-relational data of genes, miRNAs and pathways. Brief Bioinform 22:1984–1999. https://doi.org/10.1093/bib/bbz167
46. Jiang L, Zheng J, Kwan JSH et al (2019) WITER: a powerful method for estimation of cancer-driver genes using a weighted iterative regression modelling background mutation counts. Nucleic Acids Res 47:E96. https://doi.org/10.1093/NAR/GKZ566
47. Schulte-Sasse R, Budach S, Hnisz D, Marsico A (2021) Integration of multiomics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms. Nat Mach Intell 3:513–526. https://doi.org/10.1038/s42256-021-00325-y
48. Peng W, Tang Q, Dai W, Chen T (2022) Improving cancer driver gene identification using multi-Task learning on graph convolutional network. Brief Bioinform 23:432. https://doi.org/10.1093/bib/bbab432
49. Wei PJ, Wu FX, Xia J et al (2020) Prioritizing cancer genes based on an improved random walk method. Front Genet 11:377. https://doi.org/10.3389/fgene.2020.00377
50. Chaudhary MS, Pham VVH, Le TD (2021) NIBNA: A network-based node importance approach for identifying breast cancer drivers. Bioinformatics 37:2521–2528. https://doi.org/10.1093/bioinformatics/btab145
51. Pham VVH, Liu L, Bracken CP et al (2020) DriverGroup: A novel method for identifying driver gene groups. Bioinformatics 36:I583–I591. https://doi.org/10.1093/bioinformatics/btaa797
52. Pham VVH, Liu L, Bracken CP et al (2021) pDriver: a novel method for unravelling personalized coding and miRNA cancer drivers. Bioinformatics 37:3285–3292. https://doi.org/10.1093/bioinformatics/btab262
53. Guo WF, Yu X (2021) Performance assessment of sample-specific network control methods for bulk and single-cell biological data analysis. PLoS Comput Biol 17:1–32. https://doi.org/10.1371/journal.pcbi.1008962
54. Sondka Z, Bamford S, Cole CG et al (2018) The COSMIC Cancer Gene Census: describing genetic dysfunction across all human cancers. Nat Rev Cancer 18:696–705. https://doi.org/10.1038/s41568-018-0060-1
55. Xu K, Hu W, Leskovec J, Jegelka S (2019) How Powerful are Graph Neural Networks? In: International Conference on Learning Representations
56. Fan X, Gong M, Xie Y et al (2020) Structured self-attention architecture for graph-level representation learning. Pattern Recognit 100:107084. https://doi.org/10.1016/j.patcog.2019.107084
57. Xu Y, Wang J, Guang M et al (2022) Multistructure graph classification method with attention-based pooling. IEEE Trans Comput Soc Syst. https://doi.org/10.1109/TCSS.2022.3169219
58. Ying Z, Bourgeois D, You J, et al (2019) GNNExplainer: Generating Explanations for Graph Neural Networks. In: Wallach H, Larochelle H, Beygelzimer A, et al (eds) Advances in Neural Information Processing Systems. Curran Associates, Inc.
59. Yuan H, Yu H, Wang J, et al (2021) On Explainability of Graph Neural Networks via Subgraph Explorations. In: Meila M, Zhang T (eds) Proceedings of the 38th International Conference on Machine Learning. PMLR, pp 12241–12252

60. Luo D, Cheng W, Xu D et al (2020) Parameterized explainer for graph neural network. In: Ranzato M, Hadsell R et al (eds) Larochelle H. Systems. Curran Associates Inc, Advances in Neural Information Processing, pp 19620–19631
61. Sherman BT, Hao M, Qiu J et al (2022) DAVID: a web server for functional enrichment analysis and functional annotation of gene lists (2021 update). Nucleic Acids Res 50:W216–W221. https://doi.org/10.1093/nar/gkac194
62. Huang DW, Sherman BT, Lempicki RA (2009) Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. Nat Protoc 4:44–57. https://doi.org/10.1038/nprot.2008.211
63. Morris C, Ritzert M, Fey M et al (2019) Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. Proceed AAAI Conf Artif Intell 33:4602–4609. https://doi.org/10.1609/aaai.v33i01.33014602
64. Khasahmadi AH, Hassani K, Moradi P, et al (2020) Memory-Based Graph Networks. International Conference on Learning Representations
65. Fey M, Lenssen JE (2019) Fast Graph Representation Learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds
66. Mesquita D, Souza A, Kaski S (2020) Rethinking pooling in graph neural networks. Adv Neural Inf Process Syst 33:2220–2231
67. Jin T, Nguyen ND, Talos F, Wang D (2021) ECMarker: Interpretable machine learning model identifies gene expression biomarkers predicting clinical outcomes and reveals molecular mechanisms of human disease in early stages. Bioinformatics 37:1115–1124. https://doi.org/10.1093/bioinformatics/btaa935
68. Roy S, Kumar R, Mittal V, Gupta D (2020) Classification models for Invasive Ductal Carcinoma Progression, based on gene expression data-trained supervised machine learning. Sci Rep 10:4113. https://doi.org/10.1038/s41598-020-60740-w
69. Ma B, Meng F, Yan G et al (2020) Diagnostic classification of cancers using extreme gradient boosting algorithm and multi-omics data. Comput Biol Med 121:103761. https://doi.org/10.1016/j.compbiomed.2020.103761
70. del Giudice M, Peirone S, Perrone S et al (2021) Artificial intelligence in bulk and single-cell RNA-sequencing data to foster precision oncology. Int J Mol Sci 22:4563. https://doi.org/10.3390/ijms22094563
71. Wu M, Wu Z, Rosenthal DT et al (2010) Characterization of the roles of RHOC and RHOA GTPases in invasion, motility, and matrix adhesion in inflammatory and aggressive breast cancers. Cancer 116:2768–2782. https://doi.org/10.1002/cncr.25181
72. Privat M, Cavard A, Zekri Y et al (2020) A high expression ratio of RhoA/RhoB is associated with the migratory and invasive properties of basal-like Breast Tumors. Int J Med Sci 17:2799–2808. https://doi.org/10.7150/ijms.43101
73. Kalpana G, Figy C, Yeung M, Yeung KC (2019) Reduced RhoA expression enhances breast cancer metastasis with a concomitant increase in CCR5 and CXCR4 chemokines signaling. Sci Rep 9:16351. https://doi.org/10.1038/s41598-019-52746-w
74. McGranahan N, Favero F, de Bruin EC et al (2015) Clonal status of actionable driver events and the timing of mutational processes in cancer evolution. Sci Transl Med 7:283ra54-283ra54. https://doi.org/10.1126/scitranslmed.aaa1408
75. Fanjul-Fernández M, Quesada V, Cabanillas R et al (2013) Cell–cell adhesion genes CTNNA2 and CTNNA3 are tumour suppressors frequently mutated in laryngeal carcinomas. Nat Commun 4:2531. https://doi.org/10.1038/ncomms3531
76. Wen Y, Lin A, Zhu W et al (2021) Catenin Alpha-2 mutation changes the immune microenvironment in lung adenocarcinoma patients receiving immune checkpoint inhibitors. Front Pharmacol 12:645862. https://doi.org/10.3389/fphar.2021.645862
77. Hou H, Yu X, Cong P et al (2019) Six2 promotes non–small cell lung cancer cell stemness via transcriptionally and epigenetically regulating E-cadherin. Cell Prolif 52:e12617. https://doi.org/10.1111/cpr.12617
78. Liu Q, Li A, Tian Y et al (2016) The expression profile and clinic significance of the SIX family in non-small cell lung cancer. J Hematol Oncol 9:119. https://doi.org/10.1186/s13045-016-0339-1
79. Liu Z, Mar KB, Hanners NW et al (2019) A NIK–SIX signalling axis controls inflammation by targeted silencing of non-canonical NF-κB. Nature 568:249–253. https://doi.org/10.1038/s41586-019-1041-6
80. Chakravarthy R, Mnich K, Gorman AM (2016) Nerve growth factor (NGF)-mediated regulation of p75NTR expression contributes to chemotherapeutic resistance in triple negative breast cancer cells. Biochem Biophys Res Commun 478:1541–1547. https://doi.org/10.1016/j.bbrc.2016.08.149
81. Jiang M, Qin C, Han M (2016) Primary breast cancer induces pulmonary vascular hyperpermeability and promotes metastasis via the VEGF–PKC pathway. Mol Carcinog 55:1087–1095. https://doi.org/10.1002/mc.22352
82. Gottlin EB, Bentley RC, Campa MJ et al (2011) The Association of intratumoral germinal centers with early-stage non-small cell lung cancer. Journal of Thoracic Oncology 6:1687–1690. https://doi.org/10.1097/JTO.0b013e3182217bec
83. Li J, Sun R, Tao K, Wang G (2011) The CCL21/CCR7 pathway plays a key role in human colon cancer metastasis through regulation of matrix metalloproteinase-9. Dig Liver Dis 43:40–47. https://doi.org/10.1016/j.dld.2010.05.013
84. Goïta AA, Guenot D (2022) Colorectal cancer: the contribution of CXCL12 and its receptors CXCR4 and CXCR7. Cancers (Basel) 14:1810. https://doi.org/10.3390/cancers14071810
85. Li X, Wang X, Li Z et al (2019) Chemokine receptor 7 targets the vascular endothelial growth factor via the AKT/ERK pathway to regulate angiogenesis in colon cancer. Cancer Med 8:5327–5340. https://doi.org/10.1002/cam4.2426
86. Tanvir RB, Mondal AM (2020) Stage-Specific Co-expression Network Analysis for Cancer Biomarker Discovery. In: 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). pp 1813–1819
87. Su X, Xue S, Liu F et al (2022) A Comprehensive survey on community detection with deep learning. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3137396
88. Cao C, Wang J, Kwok D et al (2022) webTWAS: a resource for disease candidate susceptibility genes identified by transcriptome-wide association study. Nucleic Acids Res 50:D1123–D1130. https://doi.org/10.1093/nar/gkab957