Applied Soft Computing 135 (2023) 110040

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

EGNN: Graph structure learning based on evolutionary computation helps more in graph neural networks

Zhaowei Liu[1], Dong Yang[1], Yingjie Wang[*], Mingjie Lu, Ranran Li

Yantai University, YanTai, 264005, ShanDong, China

ARTICLE INFO

Article history:
Received 18 October 2022
Received in revised form 31 December 2022
Accepted 17 January 2023
Available online 20 January 2023

Keywords:
Graph neural networks
Evolutionary computation
Graph representation learning
Graph structure learning

ABSTRACT

In recent years, graph neural networks (GNNs) have been successfully applied in many fields due to their characteristics of neighborhood aggregation and have achieved state-of-the-art performance. While most GNNs process graph data, the original graph data is frequently noisy or incomplete, resulting in suboptimal GNN performance. In order to solve this problem, a Graph Structure Learning (GSL) method has recently emerged to improve the performance of graph neural networks by learning a graph structure that conforms to the ground truth. However, the current strategy of GSL is to iteratively optimize the optimal graph structure and a single GNN, which will encounter several problems in training, namely vulnerability and overfitting. A novel GSL approach called evolutionary graph neural network (EGNN) has been introduced in this work in order to improve defense against adversarial attacks and enhance GNN performance. Unlike the existing GSL method, which optimizes the graph structure and enhances the parameters of a single GNN model through alternating training methods, evolutionary theory has been applied to graph structure learning for the first time in this work. Specifically, different graph structures generated by mutation operations are used to evolve a set of model parameters in order to adapt to the environment (i.e., to improve the classification performance of unlabeled nodes). An evaluation mechanism is then used to measure the quality of the generated samples in order to retain only the model parameters (progeny) with good performance. Finally, the progeny that adapt to the environment are retained and used for further optimization. Through this process, EGNN overcomes the instability of graph structure learning and always evolves the best progeny, providing new solutions for the advancement and development of GSL. Extensive experiments on various benchmark datasets demonstrate the effectiveness of EGNN and the benefits of evolutionary computation-based graph structure learning.

1. Introduction

Deep learning [1] has produced significant advancements in natural language processing [2], computer vision [3], and other critical domains [4] in recent years, and it is mostly applied to data in Euclidean space. In addition, the real world contains a substantial amount of non-Euclidean graph data, such as social networks, citation networks, chemical molecular interaction networks, and knowledge graphs. Deep learning has found significant success with data in Euclidean spaces, but it performs poorly with graph data. Recent GNNs [5] aggregate feature information from local neighborhoods in each convolution layer by successfully applying the rules of neighborhood message passing [6] to graph data. Existing GNNs have done the best at a wide range of graph analysis tasks, such as classifying nodes [7,8], predicting links [9–11], and many others [12–14].

Existing GNNs have been used effectively for numerous scenarios, but they rely on the assumption that the original graph structure is real. Notably, this assumption is frequently invalid, and the quality of the original graph structure is frequently untrustworthy. To begin with, the original graph is not always owned. For instance, in computer vision [15,16], it is typically generated using subjective prior knowledge. For another, the original graph is typically noisy, and data loss is an unavoidable issue. In the chemical molecular interaction network, for instance, the edges connecting nodes are primarily derived from experimental data collected in the laboratory, and experimental errors tend to distort the reality of the network. A crucial step in GNNs is the aggregation of features. The neighborhood's feature information is aggregated and given to the neighborhood, and then the learned node embeddings are implanted for downstream tasks. It has been demonstrated that the performance of GNNs

* Corresponding author.
E-mail addresses: lzw@ytu.edu.cn (Z. Liu), yangdong@s.ytu.edu.cn (D. Yang), wangyingjie@ytu.edu.cn (Y. Wang), lumingjie@s.ytu.edu.cn (M. Lu), lrr.ytu@gmail.com (R. Li).

1 Co-first authors, contributed equally to this work.

https://doi.org/10.1016/j.asoc.2023.110040
1568-4946/© 2023 Published by Elsevier B.V.

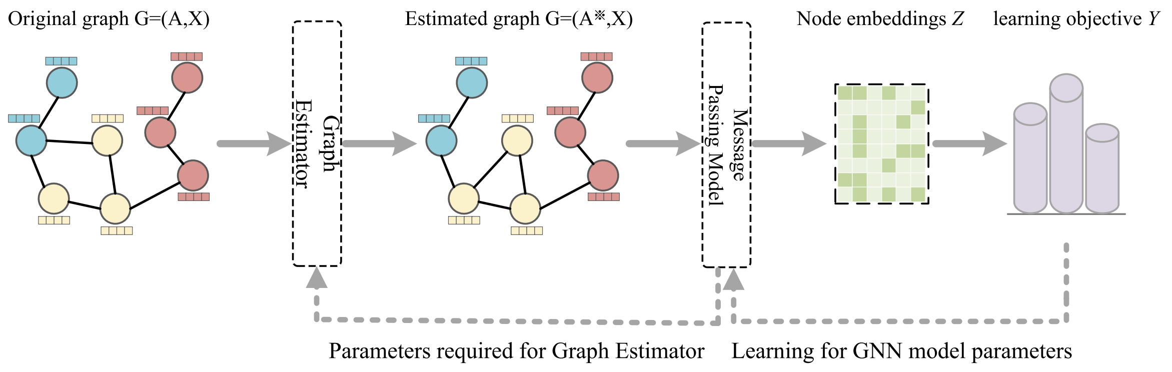


Fig. 1. A common framework for graph structure learning (GSL).

on untrustworthy graphs is far from optimal [17–19]. The performance of the same GNN on graphs with varied homogeneity (i.e., the adjacency of nodes typically belongs to the same category) is typically rather distinct. In brief, the original graph is rife with nonsensical and erroneous edges, which violates the neighborhood aggregation condition of GNNs and degrades their performance.

Graph structure learning (GSL) has received considerable attention in order to construct the ideal graph structure. As shown in Fig. 1, the goal of GSL is to learn the best graph structure and the parameters of the GNN by going through a series of steps until a predetermined stopping condition is met. Existing GSL approaches can be split mostly into two types. One is based on a single view, and the original graph is forced to be changed into an optimal structure by imposing specific constraints. Among these are: [20] promotes the training of graph neural networks by relocating the influence of labeled nodes through neighbor consistency scores; [17] generates new graph structures by learning the discrete probability distribution of connected edges on the graph; [21] enhances the model's generalization capability by removing redundant and task-independent edges from the graph; Pro-GNN [22] adds constraints, such as low rank and feature smoothness, to obtain an ideal graph structure; [23] proposes a memory-based message passing method that decouples the messages of each node into self-embedded parts for identification and memory parts for propagation and proves its effectiveness in data sets with different homogeneity ratios. The second is to reduce bias by fusing information from multiple perspectives and comprehensively estimating the optimal graph structure based on information from multiple perspectives, such as [24] based on Bayesian reasoning and incorporating various observational data into Bayesian models to reduce bias. [25] pulls out multiple basic views from the original structure and uses different information to decide on the best graph structure. [26] proposes a structure-bootstrapping contrastive learning framework that optimizes the topology of the graph through the data itself, without any labels. [27] proposes to quantify the neighborhood identity with Von Neumann enterprise and proposes the CAGNNs framework, which enhances the performance of GNNs on heterophily datasets by learning the neighbor effect of each node. However, existing methods still face the following challenges in reality: (1) Learning GNN model parameters from one information source will inevitably lead to bias. If the GNN model parameters are learned separately under multiple measurement conditions, the optimal solution is chosen based on principle, which makes the results more stable. (2) The judgment of each view in multiple views mainly depends on the discriminator and hyperparameter settings. If the view that is most beneficial to the GNN model can be determined without introducing additional mechanisms, this can reduce the complexity of the model, thereby reducing the problem of overfitting.

In order to solve these problems, this work studies a new graph structure learning paradigm, i.e., the graph structure estimator uses globally optimal node embedding feature to estimate the graph structure. In the proposed learning paradigm, the GNN model is copied into multiple models that are trained under different graph structures, and then the optimal GNN model is selected. In this case, a very natural question arises: *How do we choose the optimal GNN model in a principled way in this graph structure learning paradigm?* In order to solve these problems, this work creates a graph embedding model (EGNN) based on an evolutionary framework and treats the learning process of the optimal model parameters as an evolutionary problem in order to search for the best model parameters to improve the quality of the node embedding feature. Specifically, the performance metrics of GNNs serve as the environment (i.e., enhance the classification performance on unlabeled nodes), and the graph structure estimator evolves the population of GNN model parameters to adapt to the environment. Different graph structures can guide the parameters of the GNN model to produce different variations because varied graph structure estimators try to describe optimal graph structures from multiple perspectives, and various graph structures enable GNN models to create different model parameters to capture corresponding local structure information. As producers in the proposed model, the population of model parameters generates several variations depending on different graph structures in order to produce environmentally adapted offspring. Using the best classification performance of the present GNN, the quality of every mutant offspring is then evaluated. According to the "survival of the fittest" principle, the underperforming offspring are eliminated, while the well-performing offspring (i.e., producers) are maintained and employed for further optimization.

Based on evolutionary algorithms to optimize model parameters, the proposed EGNN overcomes the limitation of instability of model parameters trained in multiple views and always retains the optimal offspring (model parameters) generated by different graph structure generators (i.e., mutations) by way of mutations. Thus, we contribute to the development of GSL techniques.

To sum up, the contributions of this work are summarized in three aspects:

(1) A GNN model parameter evolution problem is studied within an evolutionary framework. The goal is to improve the quality of node embeddings by optimizing model parameters. The evolution process adheres to the principle of survival of the fittest and steadily improves the quality of node embedding through "mutation-evaluation" of model parameters.
(2) It is the first time that a new GSL framework based on evolution theory has been proposed. The graph structure estimator estimates the graph structure from the features

of the nodes, uses different graph structure estimators as mutation operations to create model parameter populations, and evaluates the best individual from the mutation progeny to fit the environment.

(3) By conducting extensive tests on multiple challenging datasets, EGNN's ability to achieve a desirable performance is demonstrated. In addition, several significant characteristics of EGNN are investigated.

The remainder of this paper is organized as follows. In Section 2, some related works are reviewed. In Section 3, preliminaries are introduced. In Section 4, the proposed EGNN model is qualitatively analyzed. In Section 5, the proposed model is quantitatively analyzed through experiments, and the work is concluded in Section 6.

2. Related work

According to the research topic of this work, this section provides a quick overview of the most pertinent studies on graph neural networks and graph topology learning, followed by a discussion of evolutionary algorithms for deep learning.

2.1. Graph neural networks

Graph neural networks [28,29] fall into two broad categories: graph convolution based on spectral decomposition and spatial graph convolution based on spatial transformation. The approach based on spectral decomposition can be understood as a node representation learning method based on graph spectral theory, with [30] first proposing a graph convolution network based on the Laplacian spectrum of graphs. [31] uses Chebyshev polynomials to calculate the feature matrix in order to improve computing performance. [5] introduces a semi-supervised GCN node classification model and enhances its accuracy and learning capacity by increasing the network's depth and decreasing the neighborhood width. Spatial graph convolution aggregates node neighborhood features extracted from the spatial topology of graphs; [32] performs inductive graph convolution by aggregating the neighbor features generated from multiple iterations. When aggregating neighbor nodes, [33] assigns a weight factor to each neighbor based on its characteristics. [34] presents a graph attention model for multi-label learning to improve multi-label classification performance and interpretability. [35] offers an attention-based temporal graph neural network and a spatial graph neural network, which adaptively assign the interaction weights of graph nodes in the spatial and temporal dimensions using the attention mechanism. We direct the reader to the most recent reviews [36,37] for a more exhaustive overview of graph neural networks. However, the vast majority of these graph neural networks treat the original graph structure as information that corresponds to the ground truth, severely limiting their ability to communicate uncertainty in the graph structure.

2.2. Graph structure learning

GNNs are constrained by the quality of the input graph structure, and efforts have been made to circumvent this limitation [19,21,24]. In addition to these efforts, [25] suggests a minimum sufficient graph structure. By including requirements such as low rank and feature smoothness, [38] achieves an ideal graph structure. In [10], a framework for optimum graph structure learning on heterophily networks is proposed for different graph types. [39] proposes a framework for learning the best graph structure on heterogeneous graphs. [40] describes a GaN-based approach for generating directed graphs that creates source and target nodes of nodes through joint learning.

This paper's objective is not to examine a single GNN model parameter but rather to study the evolutionary model parameter population based on graph structure learning, to identify the individuals who adapt to the environment as the parent, and the parent continues to evolve new populations. In this manner, it is possible to develop the model parameters and optimize the graph structure estimator until the ideal combination is formed, resulting in the best model.

2.3. Evolutionary algorithms for GNNs

In past research, evolutionary computing-related methods have been widely used for computational tasks such as network modeling, parameter optimization, and component design [41–44]. An evolutionary algorithm is a mature global optimization method with great robustness and broad applicability that is inspired by natural evolution. It has self-organization, self-adaptation, and self-learning properties. Individuals in a population produce offspring via mutation and choose optimal solutions based on fitness [45].

Recently, evolutionary algorithms have been developed to overcome challenges in deep learning. Numerous attempts have been made to optimize deep learning hyperparameters and construct deep network topologies [46–48] via evolutionary search in order to minimize human participation in designing deep algorithms and automatically uncover optimal configurations. Evolutionary algorithms have also been shown to be capable of optimizing deep neural networks [49–51]. In addition, Genetic-GNN [18] combines evolutionary learning with the selection of model structures and hyperparameters to dynamically approach each other's optimal fit for deep learning on graph data. [52] offers an approach to dissect changes in information flow in networks by employing evolutionary graphs to explain variations in GNN predictions. [53] presents a new GNN component automated learning framework based on evolutionary computation in order to dynamically match the optimal combination of GNN structure and model parameters.

This work is the first to explore the application of evolutionary algorithms to graph-structured learning and consider the learning process of optimal model parameters as an evolutionary problem. In addition, unlike existing methods that merely optimize the parameters of a single GNN model, graph structure learning is proposed to generate a population of GNN model parameters, with the population predicted to gradually adapt to its environment.

3. Preliminary knowledge

To support the proposed EGNN model, this section provides a brief overview of the relevant prior knowledge.

3.1. Graph embedding

The target graph can be written as $G = (A, X)$, where $A \in \mathbb{R}^{n\times n}$ is a symmetric adjacency matrix consisting of n unique nodes and expressing a sequence of edges between nodes; $A_{ij} = 1$ if there is an edge between node i and node j, otherwise $A_{ij} = 0$. $X = \left[x_1, x_2, \dots, x_N\right] \in \mathbb{R}^{N\times D}$ is the d-dimensional feature matrix of n distinct nodes, so x_i is the feature vector of node i. The objective of graph embedding is to learn a mapping $f : G \rightarrow \{h_1, h_2, \dots, h_n\}$ from the input graph, where $h_i \in \mathbb{R}^{d_l}$ is the low-dimensional vector representation of nodes, $d_l \ll d$ is the dimension of the embedding vector, and f can be a feature aggregation model whose parameters can be learned. In this work, the mapping f is accomplished by means of semi-supervised learning, i.e., in a given graph, only a small proportion of nodes $\mathcal{V}_L = \left\{v_1, v_2, \dots, v_l\right\}$ are associated with the matching labels $Y_L = \left\{y_1, y_2, \dots, y_l\right\}$, where the label of v_i is y_i.

3.2. Homogeneity of graphs

Common graph data (citation networks and knowledge graphs) conform to the property of homogeneity, i.e., nodes usually have edges with nodes of the same category, and their features are similar. However, in the real world, a considerable amount of graph data contains noise. For example, two people with different preferences in a social network may communicate, which shows that the nodes connected to each other do not belong to the same category of preferences, which can be interpreted as low homogeneity in the social network. It is worth noting that heterogeneous graphs do not necessarily have poor homogeneity of graph structures, as they depict graphs with different types (more than 2) of nodes and edges.

[18] offers the homogeneity ratio $H(g)$ to assess the homogeneity index of the graph, taking into account the homogeneity of the node's i-order neighborhood and represented as:

$$H(g) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{\left| \sum_{u \in N_{i(v)}} (y_u = y_v) \right|}{|N_i(v)|}. \tag{1}$$

The larger the value of the homogeneity ratio $H(g) \in [0, 1]$, the greater the homogeneity of the graph structure, and the smaller the value, the lower the homogeneity.

3.3. Traditional GNNs

The majority of GNNs adhere to the neighborhood aggregation paradigm [5], in which node features are updated by the node itself and its neighborhood features. The expression for neighborhood aggregation in Layer-l graph convolutional networks is:

$$h_v^l = f\left(\varphi\left(h_u^{l-1} \mid u \in N(v)\right)\right), \tag{2}$$

where h_v^l represents the hidden feature of the ith graph convolutional network, $h_v^0 = X$, $f(\cdot)$ is the conversion function between two graph convolutional layers, describing the change of the node between each layer, and $\varphi(\cdot)$ is the neighborhood aggregation function of the node that describes how features are transferred between nodes. A classic graph convolutional network aggregates and updates nodes in the following ways:

$$h_v^l = \sigma\left(W \sum_{u \in N(v)} \frac{h_u^{l-1}}{\sqrt{d_u d_v}}\right), \tag{3}$$

where W is the weight matrix applied to each node's linear transformation, d is the node's degree determined from the adjacency matrix $A + I$, $\frac{1}{\sqrt{d_u d_v}}$ is the weight between nodes u and v, and σ is the nonlinear activation function. In order to prevent overfitting, the graph convolutional layer typically has two layers. Due to the nature of neighborhood aggregation, graph convolutional network tend to produce suboptimal solutions on graphs with low homogeneity.

Based on the attention graph convolutional network, the following formula is used to define the weights between nodes:

$$\alpha_{uv} = a\left(Wh_u \parallel \mathrm{W}h_v\right), \tag{4}$$

where a is the feedforward layer parameter vector and $\parallel$ indicates the concatenation operation. According to the attention score, the significance of a node's neighbors can be determined, hence reducing graph noise to some extent.

3.4. Motivation of EGNN

Based on the issue that conventional GNNs perform poorly on graphs with poor homogeneity, we present the EGNN model, which is an evolutionary framework-based graph neural network. GNN models variation in various graphs, approaching the learning of optimal model parameters as an evolutionary problem and obtaining superior performance. Following is a description of the proposed model.

4. Model description

In this section, the problem is first defined. Then, feature extraction modules and evolutionary algorithms are introduced. The advantages of the suggested model are elaborated upon by exhibiting the mutation mechanism and EGNN evaluation. Finally, the entire EGNN training process is introduced.

4.1. Problem definition

This work focuses on the semi-supervised node classification challenge, and its objective is to steadily increase the quality of graph embeddings based on an evolutionary framework, satisfying the requirements of GNNs to develop better and more robust node embeddings. The task of "GNN parameter evolution learning" is as follows.

For the GNN parameter evolution learning task of this work, the GNN parameter space $\Theta = \left\{\Theta_1, \Theta_2, \ldots, \Theta_{E_p}\right\} \in \mathbb{R}^{|\Theta_1| \times |\Theta_2| \times \cdots \times \left|\Theta_{E_p}\right|}$ mutates individuals with different characteristics in various graph structures, where E_p is the number of model parameters in the GNN parameter space, and $\Theta_{i=1,2,\ldots,E_p} \in \mathbb{R}^{|\Theta_i|}$ is the candidate in the GNN parameter set and represents a potential solution in the parameter space. The parental mutation produces several children with varying traits, and the optimal individual (i.e., Θ_{best}) is tested for environmental adaptation. In conclusion, the best embedding performance is achieved by using GNN parameter evolution learning to create a robust model $f = \left\{\Theta_1^{\text{best}}, \Theta_2^{\text{best}}, \ldots, \Theta_{E_p}^{\text{best}}\right\}$.

4.2. Feature extraction module

The feature extraction module is also known as the graph embedding module, which may effectively apply the structure knowledge of the original graph and the similarity information between nodes to downstream tasks. In this research, the traditional GCN model [5] is employed as the basis for our feature extraction module, which captures feature information about neighbors. Given the node's feature X and the adjacency matrix A, the feature of the lth layer can be represented as:

$$H^{l+1} = \sigma\left(GCN\left(H^l, A\right)\right) \in \mathbb{R}^{n \times d}, \tag{5}$$

where the GCN model in Eq. (3) is adopted as the feature aggregation rule for the proposed EGNN model, where σ is the nonlinear activation function and $Z \in \mathbb{R}^{n \times d_z}$ represents the node features of the final layer.

4.3. Evolutionary algorithms

Throughout the entire evolution procedure, multiple graph structures are initially constructed based on the graph structure estimator. Next, the module for feature extraction trains a model parameter population Θ based on different graph topologies, where each individual represents a potential optimal solution in the parameter space Θ. Each individual is assigned a fitness value that indicates how effectively it adapts to its environment. This

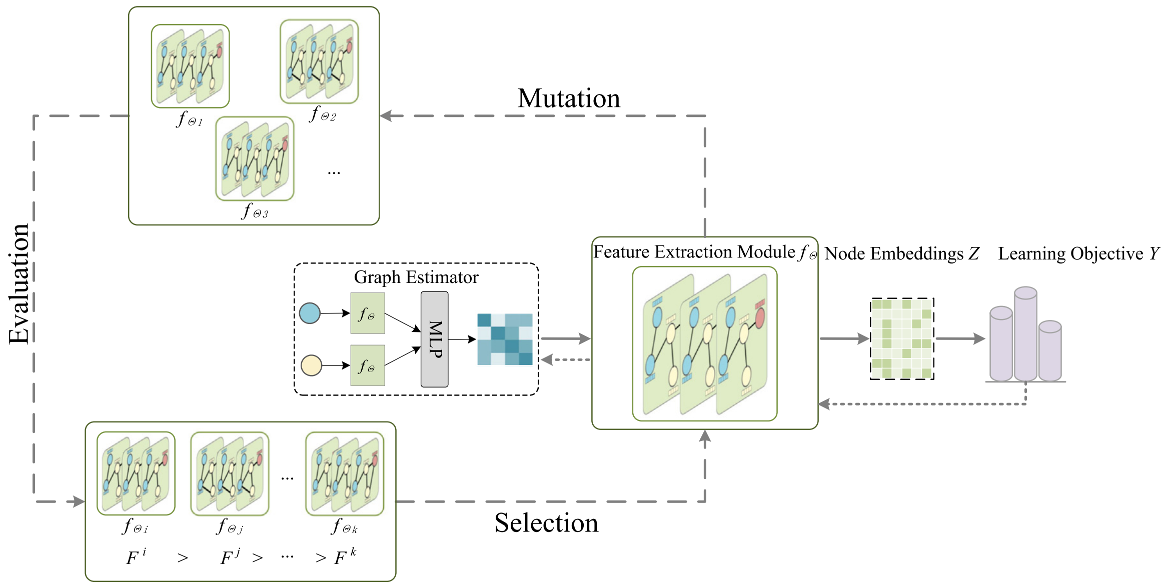


Fig. 2. The proposed EGNN model, a group of GNN models, evolves in a dynamic environment.

study anticipates that the population can progressively adapt to its surroundings, which means that the optimal option in the parameter space can create more accurate and robust features for the EGNN model, reaching state-of-the-art performance. The total framework is depicted in Fig. 2. The process of evolution can be described as follows:

1. **Initialization** (multiple graph structures)
2. **Mutations** (model parameter population)
3. **Evaluation** (model parameter population)
4. **While** (the optimal individual is incapable of environmental adaptation):
   (a) Estimate (multiple graph structures)
   (b) Mutations (population of model parameters)
   (c) Evaluation (population of model parameters)
   (d) Selection (population of model parameters)
5. **Return** the top membership of the model parameter population

where the initialization stage and the estimate stage try to estimate several graph structures using the graph structure estimator, and where each graph structure expresses the genuine adjacency connection from distinct perspectives. In the mutation step, new model parameters are generated and trained based on various graph structures. The goal is to adapt new offspring to environmental conditions through evolution. The evaluation step tries to evaluate the quality or personal attributes of each offspring based on an environment-dependent fitness function. The goal of the selection stage is to choose based on how healthy the offspring are. Only the good-quality offspring will live, become parents, and be used in the next step, training.

After each evolutionary stage, superior individuals are able to generate more reliable node embeddings, and the classification performance (environment) of unlabeled nodes is therefore updated. The environment is designed to enable individual EGNN parameters to boost the recognition performance of unlabeled nodes, expressed as:

$$\min_{\Theta, Q} \mathcal{L}(A, X, Y_L) = \sum_{v_i \in \mathcal{V}_L} \imath\left(f_\Theta(X, Q)_i, y_i\right), \tag{6}$$

where Θ is the parameter learned by EGNN, $f_\Theta(X,Q)_i$ is used to give the anticipated value of node i, and $\imath(\cdot,\cdot)$ is used to determine the gap between the expected value of the node and the actual label. The cross-entropy loss function, for instance, is used to update the model's parameters. Therefore, the environment can continuously impose new and more stringent constraints to drive the evolution of the model parameter population, resulting in improved solutions. Next, the proposed mutation operator and assessment are elaborated upon.

4.4. Mutations

Various forms of asexual reproduction are utilized to generate the next generation of individuals. In particular, these mutation operators correspond to distinct graph structures, which are re-estimated precisely by the graph structure estimator. These graph topologies are independent of one another and have distinct properties from varying perspectives, hence offering distinct information regarding the labels for the model parameters. This section introduces the mutations utilized.

4.4.1. Basic graph structure

EGNN begins by extracting various graph structures from a graph. This paper employs three widely studied fundamental graph structures: (1) the diffusion matrix A_1, which provides a global perspective of the graph structure [31]. Here, the heat kernel instance of generalized graph diffusion is used to and recalculate the first h nodes of each node, whose closed solution is $S = exp(tAD^{-1} - t)$, where t is the diffusion time parameter and D is A's degree matrix. (2) The cosine similarity-based kNN graph A_2, which depicts the similarity in the feature space, determines the edges by selecting the top k similar node pairs. The cosine value of the angle between the vectors x_i and x_j is calculated as follows:

$$S = \frac{x_i \cdot x_j}{|x_i||x_j|}, \tag{7}$$

(3) Based on the kNN graph A_3 of the heat core, the similarity is determined using Eq. (8), where t is the time parameter in the equation for heat conduction, and t is set to 2.

$$S = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{t}\right), \tag{8}$$

These three graph structures $\{A_1, A_2, A_3\}$ convey distinct features from several vantage points and serve as input to the EGNN model.

4.4.2. Graph structure estimator

After providing the three graph structures, the EGNN model re-estimates them to yield flexible graph structures that correspond to the ground truth. In this research, a graph structure estimator is used to estimate three fundamental graph structures $\{A_1, A_2, A_3\}$. First, for graph structure A_1, the node embedding H^1 is derived using Eq. (5):

$$H^1 = \sigma\left(GCN\left(X, A_1\right)\right). \tag{9}$$

The connections between the node pairs in the graph structure A_1 are then reestimated based on the node embedding H^1 learned from the graph structure A_1. Regarding node v, the embedded features are concatenated between nodes v and u, and then feed this information into the MLP layer to obtain the weight w_{vu}^1 between the nodes:

$$w_{vu}^1 = W_1 \cdot \left[H_v^1 \parallel H_u^1\right] + b_1, \tag{10}$$

where $W_1 \in R^{2d\times 1}$ represents the mapping vector and $b_1 \in R^{2d\times 1}$ represents the bias vector. Next, normalize all of the node weights to determine the confidence Q_{vu}^1 that there is an edge between nodes v and u, as follows:

$$Q_{vu}^1 = \frac{\exp\left(w_{vu}^1\right)}{\sum_{k\in s^1} w_{vk}^1}. \tag{11}$$

In order to reduce the hardware overhead, the information Q_{vu} is only calculated within a specified range S^1. For example, only the first k neighbors of the node are examined. Finally, when the graph structure A_1 is joined with the estimated graph structure V^1, the following is obtained:

$$V^1 = A_1 + \alpha^1 \cdot Q^1, \tag{12}$$

where $\alpha^1 \in (0, 1)$ represents the combination factor. Other estimated graphs $\{V^2, V^3\}$ are calculated similarly to V^1.

Given a trained classification model, the graph structure estimator's parameters are continuously tuned to define the loss function shown below:

$$\min_{\omega} \mathcal{L} = -\sum_{V\in\left\{V^1,V^2,V^3\right\}} \sum_{v_i\in\mathcal{V}_L} y_i \ln Z_i, \tag{13}$$

where Z_i is the prediction of training node v_i.

4.5. Evaluation

In the entire neural network based on an evolutionary graph, evaluation is the process of measuring individual traits or qualities. A fitness function is created to evaluate each offspring and then decide the evolutionary direction (select the optimal individual).

This section addresses two attributes: (1) personal traits and (2) qualities. When measuring a person's personal traits, the cross-entropy loss function is used to find the difference between the EGNN model's prediction Z and the true label Y. This difference is referred to as the personal traits score:

$$F_{\mathcal{L}} = -\sum_{v_i\in\mathcal{V}_L} Y_i \ln Z_i. \tag{14}$$

Notably, during training, the classification performance of unlabeled nodes is continuously enhanced to the highest score, indicating the quality of the EGNN model parameters with each iteration of evolution. If the EGNN model's prediction Z has a high score, it is highly congruent with the actual label Y.

In addition to measuring individual traits, the evaluation stage focuses on the quality of the prediction Z created by the model parameters and strive to increase the confidence of model predictions. Recently, [25] propose the importance ranking of node predictions to provide a new perspective on the ability of node predictions and to facilitate the effective training of model parameters.

This study used a similar methodology to assess the quality and stability of the predicted Z generated by the EGNN model after undergoing various mutations. There are two specific cases. First, it is assumed that the EGNN model mutates on the graph structure V^1 to generate the predicted value Z_v^1 of node v. In the first scenario, if Z_v^1 exhibits a sharp distribution (e.g., $[0.1, 0.1, 0.7, 0.1]$ for a four-item classification), whereas the other three predictors $\{Z_v^2, Z_v^3\}$ exhibit a smoother distribution (e.g., one of them behaves as $[0.25, 0.3, 0.25, 0.2]$), in instances where it is difficult for the other three models to be optimal, Z_v^1 is selected as the optimal value. In another instance in which all predicted values have the same maximum value (for example, anticipated values Z_v^2 and Z_v^3 appear as $[0.5, 0.2, 0.15, 0.15]$ and $[0.5, 0.4, 0.05, 0.05]$), it is evident that Z_v^3 is more timid than Z_v^2. In conclusion, if a predicted value has a high maximum value $Z_{v,m}$ in its classification performance and is significantly different from the next maximum value $Z_{v,nm}$, the model generating the predicted value is more likely to make confident judgments, hence receiving a higher High Quality Score. Using the aforementioned evaluation criteria, we conduct a quality evaluation of the predicted value Z learned by each EGNN model, and the quality evaluation score of prediction Z^1 is stated as follows:

$$S_{Z^1} = \sum_{v\in\mathcal{V}} e^{\gamma_1\left(\gamma_2 \log Z_{v,m}^1 + (1-\gamma_2)\log\left(Z_{v,m}^1 - Z_{v,nm}^1\right)\right)}, \tag{15}$$

where Z_v^1 is the expected value of node v on graph structure V^1, and γ_1 and γ_2 are hyperparameters. This calculation does not need the introduction of extra learning parameters, which mitigates the overfitting issue to some degree. Similarly, the quality evaluation scores of the EGNN model following mutation on the graph structures V^2 and V^3 are S_{Z^2} and S_{Z^3}. Lastly, we standardize its quality assessment score using S_{Z^1} as an example:

$$F_{Z^1} = \frac{S_{Z^1}}{S_{Z^1} + S_{Z^2} + S_{Z^3}}. \tag{16}$$

Based on the above two fitness scores, the final fitness function can be derived:

$$F = F_{\mathcal{L}} + F_Z, \tag{17}$$

higher fitness ratings for EGNN model parameters often result in more confident judgments and enhanced predictive ability.

4.6. EGNN

After describing the proposed evolutionary algorithm and its accompanying mutations and assessment, Algorithm 1 describes the EGNN's training procedure. The parameters of the GNN model are viewed as the evolutionary population, the classification performance on unlabeled nodes as the environment, the estimated different graph structures as mutation factors, and the proposed fitness function as an evaluation of whether environmental adaptation indicators exist. In each repeated evolution step, the GNN model adapts to the present environment by mutating according to various graph configurations. The "survival of the fittest" principle dictates that only well-behaved offspring survive and join future evolutionary groups. Unlike classic GNN's tyranny of learning model parameters, EGNN learns and trains many GNN model parameters depending on the graph structure, then selects the model with the greatest classification performance in order to deliver the most competitive solution. Consequently, during the training process, the evolutionary algorithm not only largely circumvents the limitation that the poor homogeneity of the original graph structure causes the GNN to be in a suboptimal solution, but also exploits the benefits of graph structure learning to seek out better solutions.

Table 1
The statistics of the datasets.

Datasets	Nodes	Edges	Classes	Features	Train	Val	Test
Polblogs	1,222	33,428	2	1,490	121	123	978
Citeseer	3,327	9,228	6	3,703	120	500	1,000
Wiki-CS	11,701	291,039	10	300	200	500	1,000
MS Academic	18,333	163,788	15	6,805	300	500	1,000
Wine	178	3,560	3	13	10	20	148
Cancer	569	22,760	2	30	10	20	539
Digits	1,797	43,128	10	64	50	100	1,647

Algorithm 1: EGNN

```
Input: original graph $A$, feature matrix $X$, labels $Y_L$,
       iterations $I$, epochs$\{E_\Theta, E_S\}$, hyper-parameter
       $\{\gamma_1, \gamma_2\}$, amount of parents $E_P$
Output: model parameters $\{\Theta^1, \Theta^2, ..., \Theta^{E_P}\}$
1  Initialization: initialize basic graphs $\{A_1, A_2, A_3\}$, initialize
     model parameters $\{\Theta_0^1, \Theta_0^2, \Theta_0^3\}$;
2  for $i = 1 : I$ do
3      for $k = 1 : E_S$ do
4          % Training Mutation Factors
5          The new graph structures are estimated according
             to equation (12);
6      end
7      for $l = 1 : E_P$ do
8          % Training Classification Models
9          for $j = 1 : E_\Theta$ do
10             Mutation model parameter $\Theta_{child}^{l,j}$ according to
                 equation (5);
11             Score $F^{l,j}$ according to equation (17);
12         end
13     end
14     Sort$\{F^{l,j}\}$;
15     $\Theta^1, \Theta^2, .., \Theta^{E_P} \leftarrow \Theta_{child}^{l_1,j_1}, \Theta_{child}^{l_2,j_2}, .., \Theta_{child}^{l_{E_P},j_{E_P}}$;
16 end
17 return model parameters $\{\Theta^1, \Theta^2, .., \Theta^{E_P}\}$;
```

5. Experiments

Extensive experiments are conducted in this section to test the performance of EGNN for semi-supervised node classification. First, baseline trials of EGNN and existing approaches are compared on several datasets, followed by the visualization of node embeddings. The graph structure estimator is then examined, and the EGNN's robustness is analyzed. The model's hyper-parameter situation is then analyzed. Finally, the complexity of the model is analyzed.

5.1. Experimental setup

5.1.1. Datasets

Evaluate seven publicly accessible benchmark datasets and classify them according to whether they are graph-type data. As shown in Table 1, use typical semi-supervised learning while assigning a different number of labels to each class in the training set.

- **Non-graph datasets**: There are three non-graph datasets available from SciKit-Learn [54]: Wine, Cancer, and Digits. Construct a preliminary adjacency matrix for non-graph datasets based on the processing of [55].
- **Academic Networks**: Both Citeseer [5] and Wiki-CS [56] are citation network datasets, with nodes representing publications and edges representing citation relationships. MS Academic [56] edges signify co-authors.
- Polblogs [22] is a blog dataset. Nodes are blog pages, edges are links from one page to another, and node labels show how political a blog is.

5.1.2. Baselines

To determine the efficacy of EGNN, it is compared to two representative GNNs, including three graph neural network algorithms (GCN [5], GAT [33], GraphSAGE [32]), and six graph structure learning methods (LDS [57], ProGNN [22], SUBLIME [26], IDGL [55], GEN [24], and CoGSL [25]). In order to be fair, experimental comparisons are based on the source code provided by the author and the way the original paper was set up.

5.1.3. Implementation details

EGNN generates four distinct basic graph structures as input for each training set. During training, the graph structure estimator selects a learning rate from 0.1, 0.01, and 0.001; the classification model uses 0.01 as the learning rate; and each layer has a separate learning rate. There is a 50% dropout rate, and the Adam optimizer is utilized. For combined coefficients, the search range for α is 0.1, 0.5, and 1.0. For hyper-parameters $\gamma_1 \in \{0.1, 0.3, 0.5, 0.9\}$, $\gamma_2 \in \{0.1, \ldots, 0.9\}$. The number of optimization iterations I is set to 200, and the parameter with the highest validation accuracy is saved for testing. For each method, five separate tests with different random seeds were done, and the average F1-macro, F1-micro, and AUC indexes were given to measure the model performance.

5.2. Node classification

The results of evaluating EGNN's performance on the semi-supervised node classification task are presented in Table 2. All datasets are divided into training, validation, and test sets and evaluated based on three standard evaluation measures. "-" in Table 2 shows that insufficient memory is available for experimental comparison. The following are our observations:

- The performance of the proposed EGNN on the seven data sets is generally better than that of other baselines, which shows the effectiveness of adding the natural evolution framework to the graph structure learning method. This is because in graph structure learning, by evolving a group of GNN model populations and selecting the optimal solution in principle, graph structure learning will be more stable, and better graph structure is more conducive to the evolution of GNN models.
- Compared with the traditional GCN, the performance of the proposed EGNN has great advantages. This phenomenon is in line with our expectation that meaningless and false edges in the original graph will prevent GCN from aggregating the correct information. This means that the proposed EGNN can estimate a more reliable graph structure to promote the learning of the GNN model to obtain more effective information.

Table 2
Node classification results (%). (Bold: best.).

Datasets	Metrics	GCN	GAT	GraphSAGE	LDS	ProGNN	SUBLIME	IDGL	GEN	CoGSL	EGNN
Polblogs	F1-macro	95.1	94.1	93.3	94.9	94.6	94.4	94.7	95.2	95.6	**95.8**
	F1-micro	95.1	94.1	93.4	94.9	94.6	93.5	94.7	95.2	95.6	**95.8**
	AUC	98.5	97.4	98.1	98.1	98.3	98.2	98.2	98.0	98.3	**98.9**
Citeseer	F1-macro	67.4	68.4	67.1	69.4	63.1	69.5	69.2	68.7	69.9	**70.3**
	F1-micro	70.1	72.2	70.1	72.2	65.6	72.5	72.6	73.0	73.4	**74.3**
	AUC	89.9	90.2	90.5	91.3	88.2	91.2	91.1	88.4	91.47	**91.5**
Wiki-CS	F1-macro	68.8	70.1	69.2	54.6	63.8	71.1	69.1	68.4	72.2	**73.2**
	F1-micro	70.8	73.8	72.2	53.7	68.3	73.4	72.7	71.1	74.5	**75.9**
	AUC	95.2	95.6	95.0	88.8	93.3	95.6	92.0	91.6	**96.4**	96.3
MS Academic	F1-macro	89.4	86.7	88.9	–	–	89.3	89.6	89.8	90.3	**90.7**
	F1-micro	91.9	89.0	91.1	–	–	90.5	91.9	92.0	**92.4**	92.2
	AUC	99.4	99.2	99.4	–	–	98.6	**99.6**	98.8	99.4	99.5
Wine	F1-macro	94.1	93.6	96.3	93.4	97.3	96.1	96.3	96.4	98.0	**98.1**
	F1-micro	93.9	93.7	96.2	93.4	97.2	95.9	96.2	96.3	97.8	**98.0**
	AUC	99.6	97.8	99.4	99.0	99.5	99.1	99.6	99.3	99.7	**99.8**
Cancer	F1-macro	93.0	92.2	92.0	83.1	93.3	93.6	93.1	94.1	94.4	**94.9**
	F1-micro	93.3	92.9	92.5	84.8	93.8	93.5	93.6	94.3	94.8	**95.4**
	AUC	**98.9**	96.9	96.9	90.6	97.8	97.6	98.1	98.3	98.3	98.4
Digits	F1-macro	89.0	89.9	87.5	79.7	89.7	90.2	92.5	91.3	92.9	**93.8**
	F1-micro	89.1	90.0	87.7	80.2	89.8	90.7	92.6	91.4	92.9	**93.9**
	AUC	98.9	98.3	98.7	95.1	98.1	98.9	99.4	98.4	99.4	**99.5**

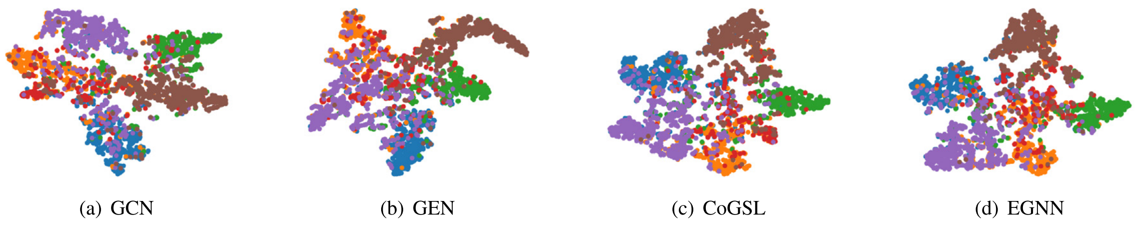

(a) GCN (b) GEN (c) CoGSL (d) EGNN

Fig. 3. A visualization of the Citeseer dataset's learned node embeddings.

- Compared with the existing graph structure learning methods, the proposed method also has good performance. This shows that the proposed evolutionary framework is helpful for learning better graph structure and GNN model parameters. Because the best GNN model and its corresponding graph structure can be judged in the evaluation steps of the proposed evolutionary framework, there is no need to introduce an additional mechanism to judge the graph structure, which reduces the problem of overfitting.

5.3. Visualization

A low-dimensional vector visualization task on the Citeseer dataset, EGNN (or GCN, GEN, CoGSL), is used as a comparison for a more intuitive comparison and to further highlight the performance of the proposed model. Specifically, the final layer of the hidden layer is output, i.e., the node embedding vector on the test set prior to the output layer, and the node embedding vector is visualized using the t-SNE method [58], as depicted in Fig. 3.

In Fig. 3, the findings of GCN, GEN and CoGSL are not always satisfying due to the fact that the degree of discriminating between nodes belonging to various categories is not evident and there are several coincidences. EGNN performs best in visualization, where the learned embeddings have a clearer structure and show that nodes from different classes are close to each other and nodes from different classes have clear borders.

5.4. Graph structure estimator analysis

The proposed EGNN has three fundamental graph structures as inputs, each of which will be re-estimated using the graph structure estimator. To test the effectiveness of the graph structure estimator described in Section 4.4.2, the model is first trained, followed by the selection of three estimated graph structures. The performance of three original graph structures, three estimated graph structures, and the final evolutionary fusion are afterwards compared. The datasets Citeseer, Wine and Digits are chosen for comparison. Fig. 4 depicts the outcomes. A_1, A_2, and A_3 are the original graph structures, whereas V_1, V_2, and V_3 are the estimated graph structures.

The following conclusions can be drawn from the experimental results: (1) All the estimated graph structures V are improved compared with the corresponding original graph structures A, which shows that the graph structures learned by the graph structure estimator are effective in helping GNN aggregate meaningful information. (2) The value of graph A_2 in Wine dataset is higher, while the value of graph A_3 in Digits dataset is higher, which indicates that using multiple graph structures is more meaningful for dealing with datasets with different characteristics. (3) EGNN is always better than the other six cases, which verifies the effectiveness of adding a natural evolution framework to the graph structure estimator. (4) Comparing the results in Fig. 4 and Table 2, it can be seen that although only a single estimated graph V is used, it still has some performance advantages over the traditional GCN model. This shows that the

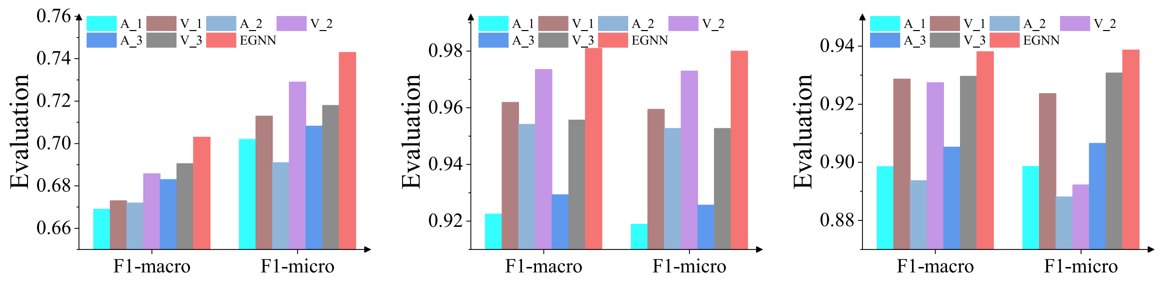
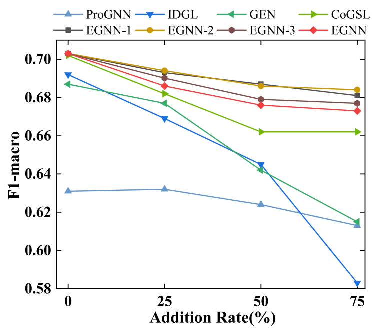
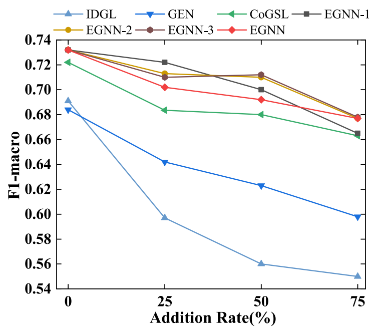


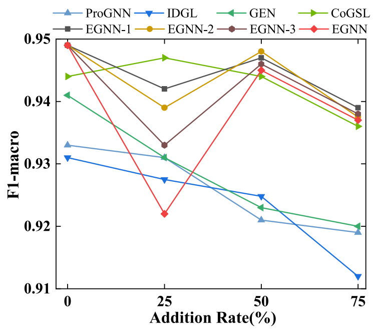
Fig. 4. Effectiveness testing of graph structure estimators.



(a) Citeseer

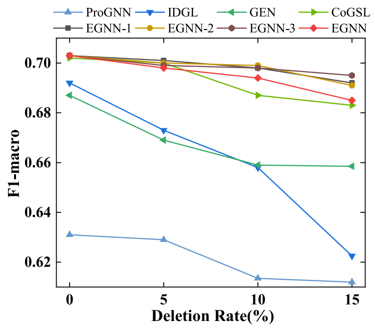


(b) Wiki-CS

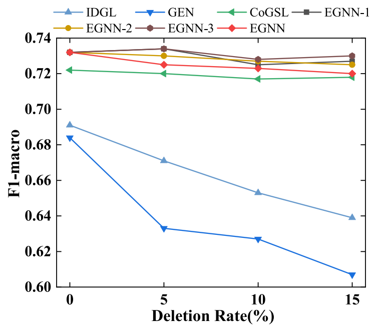


(c) Cancer

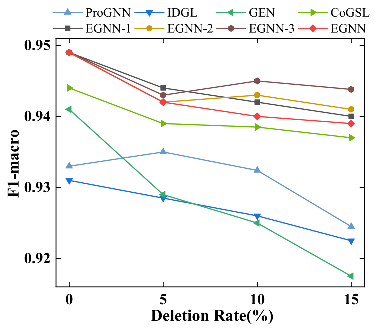
Fig. 5. Results from various models in scenarios where random edges are added.



(a) Citeseer



(b) Wiki-CS



(c) Cancer

Fig. 6. Results from various models in scenarios where random edges are deleted.

graph structure estimator plays an important role in the proposed method.

5.5. Defense analysis

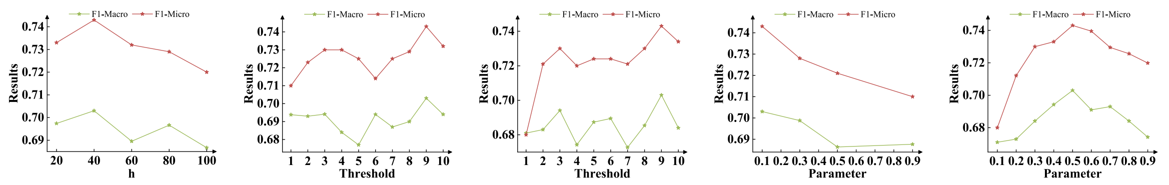
On the Citeseer, Wiki-CS and Cancer datasets, the defensive performance of many models is evaluated in this subsection. According to the strategy described in [55], the edges of the three datasets are attacked and random additions and deletions are made. Select the poison attack [59] option here. Construct an attack network on each of the three datasets before using it to train the model. GNNs that support structure learning are typically more robust than other GNNs due to their capacity to learn graph structures based on ground-truth data. Consequently, ProGNN, GEN, IDGL, and CoGSL were chosen as controls.

For adding edges, randomly add 25%, 50%, and 75% of the original number of edges to each of the three datasets (if no such edges exist). For deleting edges, 5%, 10%, and 15% of edges are randomly removed from the three datasets, but no orphan nodes are introduced. In addition, as both CoGSL and the proposed EGNN require numerous inputs at the start of training for correct evaluation, multiple inputs are attacked with the same percentage. All experiments are performed five times, and Figs. 5 and 6 display the test outcomes. In addition, EGNN-1, EGNN-2, and EGNN-3 each signal that one of EGNN's inputs is under assault, whereas EGNN implies that all three inputs are under attack.

In both instances of the three datasets, EGNN achieves superior outcomes compared to other models. The performance gap grows more obvious as the graph structure is greatly perturbed, indicating that EGNN is more robust. Also, when all of EGNN's graphs are attacked, it still does better than other GSL models. This shows that the proposed model is still effective when subjected to brute-force attacks. It can thus be concluded that the proposed model is able to defend against edge attacks.

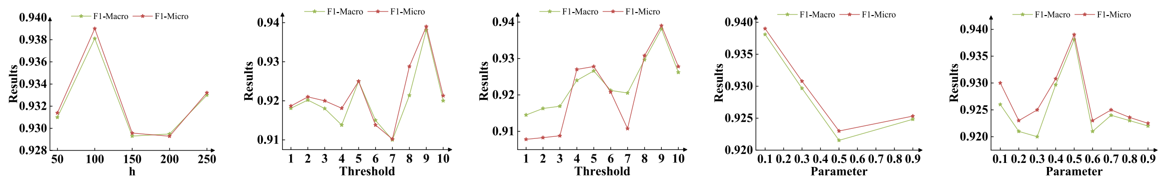
5.6. Sensitivity of hyper-parameters

In this section, the effects of the change of hyper-parameters on the Citeseer and Digits datasets on the model performance are discussed, including the first h-hop neighbor in the diffusion matrix, the coefficient k_1 in Cosine kNN, the coefficient k_2 in Heat



(a) The number of hops (b) k_1 in Cosine kNN (c) k_2 in Heat core kNN (d) Parameter γ_1 (e) Parameter γ_2

Fig. 7. Analyze the impact of the hyper-parameter range on the Citeseer dataset.



(a) The number of hops (b) k_1 in Cosine kNN (c) k_2 in Heat core kNN (d) Parameter γ_1 (e) Parameter γ_2

Fig. 8. Analyze the impact of the hyper-parameter range on the Digits dataset.

Table 3
The amount of time(s) that the models are run on the datasets.

Datasets	Polblogs	Citeseer	Wiki-CS	MS Academic	Wine	Cancer	Digits
GCN	0.0077	0.0093	0.0186	0.0285	0.0068	0.0074	0.0076
CoGSL	4.0426	8.7512	3.3483	7.488	1.0002	1.817	11.6314
EGNN	6.8452	12.849	6.7338	10.0581	1.5705	0.5429	14.0387

core kNN, and the parameters γ_1 and γ_2 in . As shown in Figs. 7 and 8, the same is true for other datasets.

The first *h*-hop neighbors in the diffusion matrix. It can be seen from the two data sets that when h is less than 100, the model performance reaches its maximum, and the maximum value of the Citeseer data set is 40. It can be seen that the reasonable selection of the range has an impact on the correct connection of the graph structure.

Thresholds k_1 and k_2. The two coefficients in the two data sets generally show an upward trend first and then a downward trend. The reason may be that when the value of k is small, information will be lost, and excessive k will introduce more noise edges.

Parameter γ_1. Verify the influence of γ_1 in . When the parameter $\gamma_1 = 0.1$ or so, the model performance reaches its best, and other values will affect the model performance.

Parameter γ_2. The effect of γ_2 is also verified in , and it is observed that with the increase of parameter γ_2, the model performance first improves and then decreases. Basically, when the parameter γ_2 is about 0.5, the model performance is optimal, and other values will damage the model performance.

5.7. Complexity

The computational complexity and execution time of the proposed method are analyzed. The complexity of each iteration within EGNN mainly involves the updating of Θ, the inference of graph structure V, and evaluation steps. Concerning the learning of parameter Θ, the optimization of all GNN models in the group can be performed in parallel, so the computational complexity is $O(E_\Theta Nd)$, where N is the number of nodes and d is the dimension of hidden nodes. As for the learning of graph structure V, the computational complexity is $O(Nd)$ since all graph structures can be calculated in parallel. The complexity of the assessment is $O(N \log N)$. In conclusion, the complexity of iteration I is about $O(IN(d(E_\Theta + 1) + \log N))$. The Table 3 depicts the average training time for each step of the proposed method and the baseline method across all data sets. This shows that mutation operations can ensure diversity in search, but the convergence speed is slower than that of traditional GSL methods. Noteworthy is the fact that the computational complexity of the proposed method can be optimized using Ball Tree [60], which can be utilized in our future work.

6. Conclusion and future work

In this paper, the application of natural evolution theory to graph structure learning is investigated for the first time, and its effectiveness is demonstrated. To implement this framework, an evolutionary algorithm is designed to evolve a population of GNN models to adapt to dynamic environments (i.e., the classification performance of unlabeled nodes). Compared with the traditional GSL method, the evolutionary paradigm can select the best offspring without introducing additional mechanisms and parameters, which improves the robustness of EGNN. Extensive experimental results confirm the effectiveness of the proposed EGNN.

Future work will investigate the interaction between graph structure learning and model parameter learning via evolutionary computation in greater depth. Intuitively, the optimal model parameters can be developed through the construction of various graph structure estimators and classification models. But figuring out how to describe the search space is still hard, and this leads to a more complicated evolutionary strategy.

CRediT authorship contribution statement

Zhaowei Liu: Conceptualization, Methodology, Writing – review & editing, Funding acquisition. **Dong Yang:** Conceptualization, Methodology, Software, Writing – original draft. **Yingjie Wang:** Conceptualization, Methodology, Writing – review & editing, Validation, Supervision. **Mingjie Lu:** Software, Data curation. **Ranran Li:** Investigation, Validation.

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

Data will be made available on request.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62272405, School and Locality Integration Development Project of Yantai City under Grant JS22JY02, the Youth Innovation Science and Technology Support Program of Shandong Provincial under Grant 2021KJ080, the Natural Science Foundation of Shandong Province under Grant ZR2022MF238, Yantai Science and Technology Innovation Development Plan Project under Grant 2021YT06000645, the Open Foundation of the State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2022-1-12.

References

[1] Geoffrey E. Hinton, Ruslan Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.
[2] Abba Suganda Girsang, et al., Modified EDA and backtranslation augmentation in deep learning models for Indonesian aspect-based sentiment analysis, Emerg. Sci. J. 7 (1) (2022) 256–272.
[3] Ahmad Aljaafreh, Ahmad Abadleh, Saqer S. Alja'Afreh, Khaled Alawasa, Eqab Almajali, Hossam Faris, Edge deep learning and computer vision-based physical distance and face mask detection system using jetson xavior NX, Emerg. Sci. J. 7 (2022) 70–80.
[4] Ricardo Costa-Mendes, Frederico Cruz-Jesus, Tiago Oliveira, Mauro Castelli, Deep learning in predicting high school grades: A quantum space of representation, Emerg. Sci. J. 6 (2022) 166–187.
[5] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the International Conference on Learning Representations, 2017.
[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl, Neural message passing for quantum chemistry, 2017, arXiv arXiv:1704.01212.
[7] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, Stefanie Jegelka, Representation learning on graphs with jumping knowledge networks, in: ICML, 2018.
[8] Keyulu Xu Weihua Hu Jure Leskovec, Stefanie Jegelka, How powerful are graph neural networks, ICLR. Keyulu Xu Weihua Hu Jure Leskovec and Stefanie Jegelka (2019).
[9] Muhan Zhang, Yixin Chen, Link prediction based on graph neural networks, Adv. Neural Inf. Process. Syst. 31 (2018).
[10] Jiaxuan You, Rex Ying, Jure Leskovec, Position-aware graph neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 7134–7143.
[11] Cheng Yang, Chunchen Wang, Yuanfu Lu, Xumeng Gong, Chuan Shi, Wei Wang, Xu Zhang, Few-shot link prediction in dynamic networks, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1245–1255.
[12] Yugang Ji, Guanyi Chu, Xiao Wang, Chuan Shi, Jianan Zhao, Junping Du, Prohibited item detection via risk graph structure learning, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1434–1443.
[13] Ranran Li, Zhaowei Liu, Yuanqing Ma, Dong Yang, Shuaijie Sun, Internet financial fraud detection based on graph learning, IEEE Trans. Comput. Soc. Syst. (2022).
[14] Shuyun Gu, Xiao Wang, Chuan Shi, Ding Xiao, Self-supervised graph neural networks for multi-behavior recommendation, in: IJCAI, 2022.
[15] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, Raquel Urtasun, 3D graph neural networks for RGBD semantic segmentation, in: 2017 IEEE International Conference on Computer Vision, 2017, pp. 5209–5218.
[16] Adam Santoro, David Raposo, David G. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, Timothy Lillicrap, A simple neural network module for relational reasoning, Adv. Neural Inf. Process. Syst. 30 (2017) 4967–4976.
[17] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, Xiao He, Learning discrete structures for graph neural networks, in: International Conference on Machine Learning, 2019, pp. 1972–1982.
[18] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, Bo Yang, Geom-GCN: Geometric graph convolutional networks, in: International Conference on Learning Representations, 2019.
[19] Yingxue Zhang, Soumyasundar Pal, Mark Coates, Deniz Ustebay, Bayesian graph convolutional neural networks for semi-supervised classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, (01) 2019, pp. 5829–5836.
[20] Ming Xu, Baoming Zhang, Hualei Yu, Jinliang Yuan, Chongjun Wang, NC-GNN: Consistent neighbors of nodes help more in graph neural networks, Wirel. Commun. Mob. Comput. 2022 (2022).
[21] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, Wei Wang, Robust graph representation learning via neural sparsification, in: International Conference on Machine Learning, 2020, pp. 11458–11468.
[22] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, Jiliang Tang, Graph structure learning for robust graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 66–74.
[23] Jie Chen, Weiqi Liu, Jian Pu, Memory-based message passing: Decoupling the message for propagation from discrimination, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2022, pp. 4033–4037.
[24] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, Xing Xie, Graph structure estimation neural networks, in: Proceedings of the Web Conference 2021, 2021, pp. 342–353.
[25] Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, Chuan Shi, Compact graph structure learning via mutual information compression, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1601–1610.
[26] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, Shirui Pan, Towards unsupervised deep graph structure learning, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1392–1403.
[27] Jie Chen, Shouzhen Chen, Zengfeng Huang, Junping Zhang, Jian Pu, Exploiting neighbor effect: Conv-agnostic GNNs framework for graphs with heterophily, 2022, arXiv arXiv:2203.11200.
[28] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, Liwei Wang, Graphnorm: A principled approach to accelerating graph neural network training, in: International Conference on Machine Learning, PMLR, 2021, pp. 1204–1215.
[29] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, Xavier Bresson, Benchmarking graph neural networks, 2020, arXiv arXiv:2003.00982.
[30] Joan Bruna, Wojciech Zaremba, Arthur Szlam, Yann LeCun, Spectral networks and deep locally connected networks on graphs, in: Proceedings of the International Conference on Learning Representations, 2014.
[31] Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016).
[32] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, Graph attention networks, in: Proceedings of the International Conference on Learning Representations, 2018.
[34] Kien Do, Truyen Tran, Thin Nguyen, Svetha Venkatesh, Attentional multilabel learning over graphs: a message passing approach, Mach. Learn. 108 (10) (2019) 1757–1781.
[35] Hao Zhou, Dongchun Ren, Huaxia Xia, Mingyu Fan, Xu Yang, Hai Huang, AST-GNN: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction, Neurocomputing 445 (2021) 298–308.
[36] Lilapati Waikhom, Ripon Patgiri, Graph neural networks: Methods, applications, and opportunities, 2021, arXiv arXiv:2108.10733.
[37] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.
[38] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, Liang Wang, Deep graph structure learning for robust representations: A survey, 2021, arXiv preprint arXiv:2103.03036.
[39] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, Yanfang Ye, Heterogeneous graph structure learning for graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, (5) 2021, pp. 4697–4705.
[40] Shijie Zhu, Jianxin Li, Hao Peng, Senzhang Wang, Lifang He, Adversarial directed graph embedding, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, (5) 2021, pp. 4741–4748.
[41] Kenneth De Jong, Evolutionary computation: a unified approach, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2017, pp. 373–388.
[42] Yi nan Guo, Xu Zhang, Dun wei Gong, Zhen Zhang, Jian-Jian Yang, Novel interactive preference-based multiobjective evolutionary optimization for bolt supporting networks, IEEE Trans. Evol. Comput. 24 (2020) 750–764.

[43] Sreenivas Sremath Tirumala, Evolving deep neural networks using coevolutionary algorithms with multi-population strategy, Neural Comput. Appl. 32 (16) (2020) 13051–13064.

[44] Roua Jabla, Maha Khemaja, Sami Faiz, Decision-making improvement in dynamic environments using machine learning, J. Human Earth Future 3 (1) (2022) 55–68.

[45] Agoston E. Eiben, Jim Smith, From evolutionary computation to the evolution of things, Nature 521 (7553) (2015) 476–482.

[46] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al., Evolving deep neural networks, in: Artificial Intelligence in the Age of Neural Networks and Brain Computing, Elsevier, 2019, pp. 293–312.

[47] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, Alexey Kurakin, Large-scale evolution of image classifiers, in: International Conference on Machine Learning, PMLR, 2017, pp. 2902–2911.

[48] Yinan Guo, Guoyu Chen, Min Jiang, Dunwei Gong, Jing Liang, A knowledge guided transfer strategy for evolutionary dynamic multiobjective optimization, IEEE Trans. Evol. Comput. (2022) 1.

[49] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, Evolving deep convolutional neural networks for image classification, IEEE Trans. Evol. Comput. 24 (2) (2019) 394–407.

[50] Sean Lander, Yi Shang, EvoAE–A new evolutionary method for training autoencoders for deep learning networks, in: 2015 IEEE 39th Annual Computer Software and Applications Conference, Vol. 2, IEEE, 2015, pp. 790–795.

[51] Junhao Huang, Weize Sun, Lei Huang, Deep neural networks compression learning based on multiobjective evolutionary algorithms, Neurocomputing 378 (2020) 260–269.

[52] Yazheng Liu, Xi Zhang, Sihong Xie, Explaining GNN over evolving graphs using information flow, 2021, arXiv preprint arXiv:2111.10037.

[53] Min Shi, David A. Wilson, Xingquan Zhu, Yu Huang, Yuan Zhuang, Jianxun Liu, Yufei Tang, Evolutionary architecture search for graph neural networks, 2020, arXiv preprint arXiv:2009.10199.

[54] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[55] Yu Chen, Lingfei Wu, Mohammed Zaki, Iterative deep graph learning for graph neural networks: Better and robust node embeddings, Adv. Neural Inf. Process. Syst. 33 (2020) 19314–19326.

[56] Johannes Klicpera, Aleksandar Bojchevski, Stephan Günnemann, Predict then propagate: Graph neural networks meet personalized PageRank, in: ICLR, 2019.

[57] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, Xiao He, Learning discrete structures for graph neural networks, in: ICML, 2019.

[58] G. Hinton, L.J.P. van der Maaten, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.

[59] Tailin Wu, Hongyu Ren, Pan Li, Jure Leskovec, Graph information bottleneck, Adv. Neural Inf. Process. Syst. 33 (2020) 20437–20448.

[60] Stephen M. Omohundro, Five Balltree Construction Algorithms, International Computer Science Institute Berkeley, 1989.