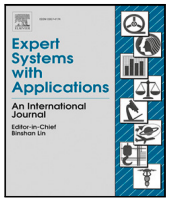
Addressing imbalance in graph datasets: Introducing GATE-GNN with graph ensemble weight attention and transfer learning for enhanced node classification

Expert Systems With Applications 255 (2024) 124602

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Addressing imbalance in graph datasets: Introducing GATE-GNN with graph ensemble weight attention and transfer learning for enhanced node classification

Abdul Joseph Fofanah [a,1,*], David Chen [a], Lian Wen [a], Shaoyang Zhang [b]

[a] School of Information and Communication Technology, Griffith University, 170 Kessels Road, Brisbane, 4111, Queensland, Australia
[b] School of Information and Communication Technology, Chang'an University, Xi'an, China

ARTICLE INFO

Keywords:
Classification
Ensemble method
Ensemble attention
Graph neural network
Transfer learning method
Imbalanced dataset

ABSTRACT

Significant challenges arise when Graph Neural Networks (GNNs) try to deal with uneven data. Specifically in signed and weighted graph structures. This makes classification tasks less effective. Within the GNN context, researchers have found traditional solutions like resampling, reweighting, and synthetic sample generation to be inadequate. GATE-GNN, a novel architecture designed specifically for imbalanced datasets, overcomes these limitations. GATE-GNN integrates an ensemble of network modules that harness the spatial features of graph networks and effectively utilise embedding information from earlier layers. This unique approach not only bolsters generalisation by reducing volatility. It also refines the optimisation algorithm, resulting in more accurate and stable classification outcomes. We rigorously tested the effectiveness of GATE-GNN on four widely recognised datasets: Cora, NELL, Citeseer, and PubMed. We performed a comparative analysis against established methods such as Graph Convolutional Networks (GCN), Graph Sample and Aggregate (GraphSAGE), Propagation Multilayer Perceptron PMLP), Imbalanced Node Sampling GNN (INS-GNN), GNN-Curriculum Learning (GNN-CL) and Graph Attention Networks (GAT). Empirical results demonstrate that GATE-GNN significantly outperforms these existing models, achieving an average improvement in classification accuracy of approximately 5%–10% over the previous best results. Additionally, GATE-GNN presents a marked reduction in training time. This underscores its efficiency and suitability for practical applications in imbalanced graph data scenarios. Implementation of the proposed GATE-GNN can be accessed here https://github.com/afofanah/GATE-GNN.

1. Introduction

By assigning labels to nodes within a graph based on which both the inherent graph structure and node attributes in Graph Neural Networks (GNNs) have emerged as a powerful tool for node classification tasks. Research on GNNs are instrumental in various domains, such as social network analysis, deciphering roles and interests through connection patterns, and communities (Maurya, Liu, & Murata, 2022; Wei, Wang, Hu, Li, & Yi, 2022). Maekawa, Noda, Sasaki, et al. (2022) contributed to categorising research papers in citation networks and predicting molecular features in molecular graph analysis.

The processing of complex and diverse graph data that are imbalanced in nature are often pervasive challenge in GNNs node classification problems. Class size distributions and edge connection proportions are factors that significantly impact GNN efficiency and resilience (Maekawa et al., 2022). According to Junior, do Carmo Nicoletti, and Zhao (2017), other factors like biassed or random distribution of attributes also affect the correlations between node characteristics and labels and consequently complicate node classification.

In order to tackle these challenges, several models and techniques have been developed. Some of these techniques include Graph Attention Networks (GATs) and Boosting Graph Neural Networks (BGNs).

* Corresponding author.
E-mail addresses: abdul.fofanah@griffithuni.edu.au (A.J. Fofanah), d.chen@griffith.edu.au (D. Chen), l.wen@griffith.edu.au (L. Wen), zhsy@chd.edu.cn (S. Zhang).
[1] First author.

https://doi.org/10.1016/j.eswa.2024.124602
Received 19 December 2023; Received in revised form 18 June 2024; Accepted 23 June 2024
Available online 28 June 2024

Guo, Xu, and Tao (2021) utilised GATs to dynamically weigh neighbouring nodes based on relevance. The BGNs sequentially train multiple GNNs by adjusting sample weights to optimise classification. Acharya and Zhang (2020) employed SoftMax and L2-normalisation for effective feature selection called Feature Selection Graph Neural Networks (FSGNNs) to address the problem of graph node classification.

Graph-of-Graph Neural Networks (G GNN) was introduced as a solution by Wang, Zhao, Shah and Derr (2022) to handle the challenges of imbalanced graph classification. Supervision from surrounding graphs on a global scale as well as from stochastic augmentations of graphs on a local scale were achieved. According to Shi et al. (2021), Boosting-GNN was developed for handling imbalanced node classification in graph networks, cost-sensitive learning (Cui, Jia, Lin, Song, & Belongie, 2019; Cui, Zhang, Noh, & Park, 2022). The study shows the Boosting-GNN can reduce computational complexity when transfer learning was employed. Reweighting and resampling method was employed by Kaur, Pannu, and Malhi (2019), Liu et al. (2022) as contribution to address the challenges of imbalanced node classification in GNN networks. The loss function was integrated to class frequency and node degree that effectively balances the node distribution and the graph structure. The application of self-training method was introduced by Juan, Peng, and Wang (2021), Wang, Xiong, Hou and Wu (2022) to address the problem of imbalanced node classification. These techniques are enhanced to balance the data distribution, adjust the loss function, or augment the dataset.

Additionally, GNN framework involves peculiar challenges especially how to select the optimal number and type of attention heads for each GNN module and how to balance the trade-off between expressiveness and complexity; how to design an effective graph ensemble weight attention mechanism that can dynamically adjust the weights of different GNN modules and attention heads according to the graph structure and the node features; how to choose the appropriate source and target domains for transfer learning, and how to align the distributions and representations of the nodes across different domains and how to evaluate the effectiveness and the generalisability of the method on various graph datasets and tasks over existing methods (Sun et al., 2023). Solving these issues is the focus of our study.

In order to address the challenges of generalisation and handling imbalanced data in GNNs, we proposed the Graph Attention Ensemble GNN, called GATE-GNN. Our method utilises weighted edges and attention mechanisms. GATE-GNN effectively learns from graph structures and significantly enhances performance on imbalanced datasets. It incorporates multiple attention heads for edge weight calculations and aggregates through an adaptive boosting technique. This approach, coupled with transfer learning, optimises the model's fitting capability while reducing computation complexities.

This study enables us to understand the pros and cons of the GAT, GCN, and MLP traditional deep learning models. Firstly, by visualising the data, identifying the most important functions of aggregating nodes, and predicting the classification problem, secondly, how to choose the best number and type of attention heads for each GNN module while balancing expressiveness and complexity; how to create a useful graph ensemble weight attention mechanism that can change the weights of various GNN modules and attention heads based on the structure and features of the nodes; and how to pick the right source and target domains for transfer learning.

The GATE-GNN's performance was benchmarked against traditional GNN models on various datasets and demonstrates its exceptional capability in managing synthetically imbalanced datasets. It achieves notable improvements in both classification accuracy and training efficiency.

The key contributions of this research are:

- The introduction of GATE-GNN, which aggregates node features efficiently using adaptable weights through the Multilayer Attention Score (MAS) and Graph Node Features Extraction (GNFE) technique, creating suitable node embeddings for classification tasks.
- Development of the Graph Ensemble Weight Attention (GEWA) as a novel self-attention mechanism, enhancing feature representation post GNFE extraction and Node Ensemble and Transfer Learning (NETL) scheme significantly reducing computational and memory demands.
- Exploration of dynamic node interactions within GATE architecture, leveraging GNFE and GEWA to apply learnable weights and attention scores, thus improving feature representation.
- Extensive empirical validation demonstrating GATE-GNN's superior performance in addressing imbalanced data challenges compared to existing methods.

The remainder of the paper is structured as follows: Section 2 delves into the background of handling imbalanced data in GNNs, providing insights into the model's generalisation strategies, covering the challenges and impacts as well as common techniques employed in the field. Section 3 explores the rationale behind the GATE-GNN model's effectiveness in managing imbalanced data and the utility of Graph Ensemble Weight Attention (GEWA). Section 4 details the proposed GATE-GNN methodology, elaborating on its architecture and functionalities. In Section 5, we present experimental results and benchmarks to evaluate the effectiveness of the GATE-GNN model. The paper concludes with Section 6, summarising the key findings and contributions of the study.

2. Background

This section delves into the complex challenges and consequential impacts associated with addressing imbalanced datasets in the context of graph neural networks (GNNs). The focus is particularly on node classification problems. We explore various common methods employed to mitigate these imbalances and examine the role of base classifiers in GNNs. The background exploration provides the foundational understanding necessary to comprehend the complexities and nuances inherent in optimising GNNs for imbalanced graph data.

2.1. Challenges and impact of handling imbalanced data

Training and evaluation processed in graph data poses significant challenges in machine learning and deep learning for GNNs imbalanced data (Wang, 2022; Zhao, Zhang, & Wang, 2021). When one class is overrepresented and causes the model to potentially exhibit bias towards the majority class to the detriment of the minority class. This is usually known as imbalanced data (Juan et al., 2023; Wang, Xiong, et al., 2022).

Key challenges resulting from imbalanced data include the following:

- **Training issues**: Apparently, a limited dataset size can hinder the model's capability to capture the relevant features of the minority class in imbalanced data. Poor generalisation and overfitting are the results of the dominant class during training.
- **Evaluation issues**: When relying on basic metrics such as accuracy, the imbalanced data can skew the model's accuracy and performance evaluation. An overestimated perception of model effectiveness can be misplaced by overlooking the misclassification issues of the minority class. To address this, perturbations need to be applied in order to determine the model's stability and adaptability in GNN classification.

Nevertheless, if we make the assumption that the classes C have a diverse range of nodes, the conventional node classifier will exhibit subpar performance (Zhao, Zhang, & Wang, 2021). The class imbalance ratio (IR) is defined as the ratio between the smallest minority class and the largest minority class in the graph network. It suggests that the original graph exhibits a class imbalance issue, with the number of

false users being significantly fewer than the number of real users. A large IR suggests that the node classification problem is skewed, and the performance of the GNN model may be adversely affected when dealing with minority classes. The objective is to develop a node classifier that demonstrates proficient classification performance for both the minority and majority classes. Additionally, it should be capable of predicting the labels of unlabelled nodes. The equation serves as a depiction of the infrared spectrum.

$$\mathbf{IR} = \frac{\max_{c \in C} |V_c|}{\min_{c \in C} |V_c|}, \tag{1}$$

where C is the set of classes, V_c is the set of nodes that belong to class c, and $|\cdot|$ denotes the cardinality of a set.

In the context of a GNN node classification task, the objective is to assign class labels to the nodes of a graph. The goal is to predict the class labels of unlabelled nodes by utilising both the graph structure and the properties of the nodes. The IR can be calculated using either the labelled nodes in the training set or the full graph.

2.2. Common techniques to handle imbalanced data

This subsection explores a range of established techniques in Graph Neural Networks (GNNs) for tackling imbalanced datasets. It also addressing their inherent limitations. Each of these methods contributes to the handling of imbalanced datasets in GNNs. However, they also bring unique challenges that must be considered for effective model development and implementation.

- **Graph convolutional technique**: Where imbalances are common, graph convolutional networks (GCNs) are capable of analysing non-Euclidean graph data. In order to enable feature extraction that considers graph topology, the convolutional features for graph data are adaptive data (Guo et al., 2022; Han, Huang, An, & Bai, 2021). Ivanov and Prokhorenkova (2021), Shi et al. (2021), Zhang, Hu, Sun, Wen, and Zhang (2023) uses a Sylvie and Boosting-GNN framework to leverage the capabilities of GCN. GCN is capable of processing large-scale graphs and addressing node classification imbalances. However, there are some challenges with the inherent noise and complexity of the graph data.
- **Resampling technique**: Resampling balances class distribution through oversampling (adding minority class samples) or undersampling (reducing majority class samples). According to Fiorentini and Losa (2020), Kaur et al. (2019), resampling techniques are effective in creating imbalanced datasets. However, it possesses noise and bias that may account for the true structure of the graph data. This may also lead to potential misrepresentation.
- **Synthetic data generation**: SMOTE and ADASYN are methods that create new minority class instances and enhance representation. It also preserves data diversity (Douzas & Bacao, 2019; Han, Wang, & Mao, 2005). The methods improve training data. However, it can be computationally expensive and introduce noise and bias.
- **Cost-sensitive learning and reweighting**: Cui et al. (2019), Lin, Goyal, Girshick, He, and Dollár (2017) introduces the assign strategy by assigning different weights to classes based on the challenges of imbalanced data. They also applied focal loss and class-balanced focal loss techniques. It requires careful parameter tuning and can complicate the optimisation process.
- **Ensemble classifier**: Ensemble methods combine multiple models to create robust predictions while effectively addressing individual model biases (Pfeifer et al., 2023; Shi et al., 2021). It enhances prediction stability and accuracy. However, it increases computational complexity and necessitates advanced evaluation techniques.

2.3. GNNs base classifier

This subsection focuses on the utilisation of graph neural networks (GNNs) in managing imbalanced data within graph-structured data. The Graph Convolutional Networks (GCNs), GraphSAGEs, and Graph Attention Networks (GATs) were carefully examined for the classification challenge. These three well-established GNN models employ distinct methods for aggregating information from neighbouring nodes.

- **Graph Convolutional Networks (GCNs)**: According to Karagiannakos (2021), the GCNs leverage the spectral graph theory for graph convolution functions. By incorporating both local structure and node features that enable it to compute node representations using linear transformations and nonlinear activations. Nevertheless, oversmoothing and oversimplification in large or deep graphs is a challenge. This limits their effectiveness in complex network analysis.
- **GraphSAGE (Graph Sample and Aggregate)**: GraphSAGE uses spatial graph theory that computes node representations through the application of nonlinear functions to concatenate nodes and aggregate neighbour features (Hamilton, Ying, & Leskovec, 2017). This model improves in capturing global correlations and node diversity. However, it faces challenges with computational cost and memory use in dense and complicated graphs.
- **Graph Attention Networks (GATs)**: According to Velickovic et al. (2017), the GATs integrate a self-attention mechanism for convolution functions in graph data. The linear transformations and nonlinear activations are computed by node representations. GATs are capable of effectively capturing long-range relationships and the relevant nodes. However, they may face challenges of high parameter complexity and instability in large and noisy networks.

The techniques and challenges of imbalanced datasets within the GNN framework and GNN base classifiers have been discussed in this section. These strategies underpin our proposed GATE-GNN model. This is subsequently elaborated on in the next section of the paper.

2.4. Why deep learning models can handle imbalanced data?

This section underscores the core principles underpinning the GNNs model. Its effectiveness in managing imbalanced data within the GNNs framework is discussed. We shed light on why GNN models stands out as an optimal methods to handle imbalanced data in GNN.

Here are GNNs generalisation strategies utilised by recent research work:

- The quest for optimal generalisation in GNNs and in the context of imbalanced data subsequently led to diverse methodological improvements. The theoretical aspects of this challenge and focusing on the complexities of graph structures were delved into by Puny, Ben-Hamu, and Lipman (2020). However, this approach grapples with computational and training complexities and skirmishes to comprehensively address imbalances. The potential for noise or bias in data when synthetic samples fail to authentically represent the distribution of the true graph structure.
- In contrast, Zhao, Qiu, Wen, Liao, and Huang (2021), Zhou, Chen, and Lin (2022) adopted the Graph Attention Network (GAT) strategy that assigns varying weights to neighbouring nodes. Additionally, they integrated a boosting method to balance sample weights and reduce bias towards dominant classes. While effective, this method introduces computational and scalability challenges. This often necessitates the generation or deletion of a large volume of samples.

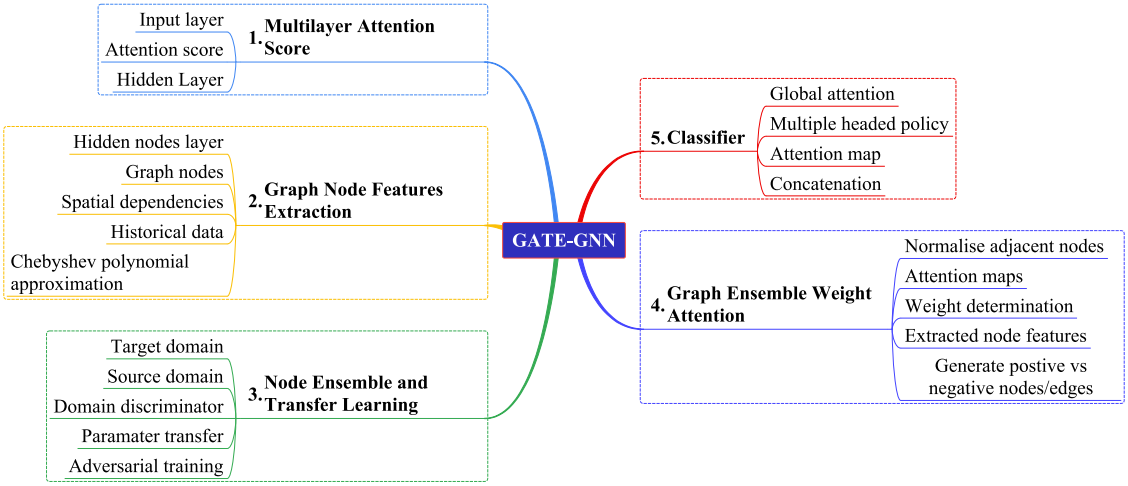


Fig. 1. The 5 modules of the propose model.

- Ensemble techniques such as bagging, Synthetic Minority Oversampling Technique (SMOTE), and random sampling were other approaches employed (Huang, Li, Loy, & Tang, 2016; Ojala, Pietikainen, & Maenpaa, 2002; Zhao et al., 2022). These techniques aim to balance data subsets within each model and enhance accuracy and model robustness. However, they also escalate computational costs and model complexity. It also necessitates advanced evaluation techniques and metrics.
- Finally, leveraged transfer learning to improve GNN performance in imbalanced node classification was introduced by Shi et al. (2021). Other recent techniques includes propagation MLP (Yang, Wu, Wang, & Yan, 2023), self-supervised learning (Juan et al., 2023), and curriculum learning (Li et al., 2024). This technique boosts adaptability across different data distributions and environments. Nevertheless, potential overfitting and the need for more complex evaluation strategies in the majority class context pose challenges within the GNN classification framework.

3. Development of the GEWA approach for addressing imbalanced data in GNNs

This part talks about the rationale for utilising ensemble weights to aggregate the most relevant information in the GNN framework, which led us to develop the GEWA approach to experiment with the work of GATE-GNN for imbalanced data.

Imbalanced data presents a formidable challenge in the field of graph neural networks (GNNs), particularly impacting the performance of node classification tasks. Traditional techniques, while valuable, often fall short in providing balanced representation across diverse classes. To bridge this gap, we have drawn inspiration from existing methodologies to conceive the Graph Ensemble Weight Attention (GEWA) approach.

The genesis of GEWA lies in a meticulous analysis of ensemble and transfer learning techniques, both of which have demonstrated their potential in enhancing GNN performance. Ensemble methods, known for their ability to amalgamate multiple model predictions, offer a pathway to leverage the collective strengths of various GNN architectures. Meanwhile, transfer learning techniques excel in applying knowledge from one domain to improve performance in another, particularly useful in dealing with sparse data in minority classes.

In synthesising these two approaches, we identified a critical opportunity: to dynamically adjust the weights of different GNN modules in response to the distinct structural and feature-related patterns present within imbalanced datasets. This realisation led us to the concept of GEWA, which employs an attention mechanism to assign importance to different GNNs based on their contribution to the classification task. By focusing on the most informative features and minimising the influence of overrepresented classes, GEWA aims to bring equilibrium to the learning process.

Our proposed GEWA method, therefore, stands as a testament to innovative thinking in the GNN landscape. It encapsulates the strength of ensemble learning while incorporating the adaptability of transfer learning. Such integration promises to address the biases inherent in imbalanced datasets, as it ensures that all classes, irrespective of their size, are given due consideration in the model's learning process.

In operationalising GEWA within our GATE-GNN model, we have established a framework that is both robust and sensitive to the nuances of graph-structured data. As we look forward to substantiating the efficacy of GEWA through empirical testing, we remain confident in its potential to revolutionise node classification in GNNs.

With the groundwork laid, the ensuing sections will delve into the specifics of the GATE-GNN method, elaborate on experimental validations, and conclude with a synthesis of our findings.

4. Proposed method

In this section, we delineate the architecture of our newly proposed model, the Graph Attention Ensemble GNN (GATE-GNN). Composed of five distinct modules, the architecture is depicted in Fig. 1. We commence with an exposition on the overarching framework of GATE-GNN, followed by an in-depth exploration of each module's functionality and interplay within the subsequent subsections.

4.1. Overall structure

The Graph Attention Ensemble GNN (GATE-GNN) is articulated through five interconnected modules, each fulfilling a pivotal role within the overarching architecture, as explicated in Fig. 1. These modules are enumerated and described as follows:

1. **Multilayer Attention Score (MAS)**: As the initial module, MAS enriches the input data by appending attention scores, prepping the data for enhanced feature processing.
2. **Graph Node Features Extraction (GNFE)**: Utilising the MAS-processed data, GNFE employs Graph Convolutional Networks (GCN) to extract essential graph features.
3. **Node Ensemble and Transfer Learning (NETL)**: NETL extends the capabilities of GNFE by capturing temporal dependencies within the data, which are vital for the input to the subsequent module.

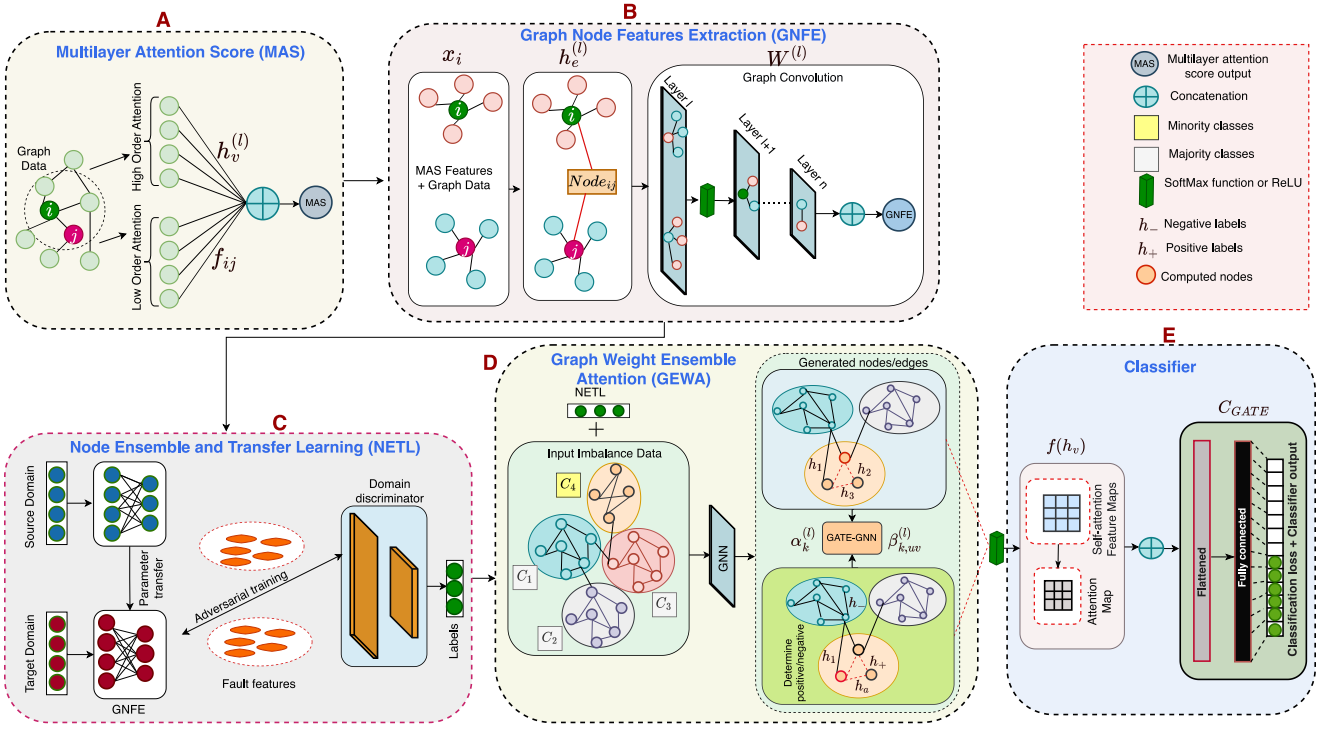


Fig. 2. Overall architecture of the proposed GATE-GNN model: (A) shows the multilayer attention score (MAS) module, where i and j are the associated features between neighbouring nodes, $h_v^{(l)}$ is the high-order attention, and f_{ij} is the low-order attention. (B) shows the graph node feature extraction (GNFE) module, where x_i is the input vector node and MAS features and $W_{(l)}$ is the weight vector of the l-layer. (C) shows the graph node ensemble and transfer learning (NETL) module using adversarial training to predict the labels. (D) shows the weight ensemble attention (GEWA) module, where $\alpha_k^{(l)}$ is the scalar weights of the GEWA, $\beta_{k,uv}^{(l)}$ is the attention coefficient, and $h_{(-+)}$ is the negative and positive labels generated. (E) shows the GATE-GNN classifier, where $f(h_v)$ is the self-attention weight classifier and C_{GATE} is the GATE-GNN classifier.

4. **Graph Ensemble Weight Attention (GEWA)**: The core of GATE-GNN, GEWA, integrates inputs from GNFE and NETL, applying spatial attention to aggregate neighbouring nodes' features.
5. **Classifier**: This final module consolidates the outputs from GEWA, adopting a multi-headed classification strategy to accurately categorise the data.

The intricate architecture of GATE-GNN, depicting the inter-module relationships and data flow, is presented in Fig. 2.

4.2. Multilayer attention score (MAS)

With the multilayer attention score, the GNN will be better able to find the connections between different levels of graph knowledge and learn more expressive and generalisable node features. The multilayer attention score is a way of measuring the importance of each node's features across different layers of the GNN framework. It is based on the idea of graph attention networks (GATs) and multilayer perceptrons (MLPs), which use a self-attention mechanism to assign different weights to the nodes and their neighbours in each layer. The multilayer attention score extends this idea by allowing the nodes to attend to different layers of the GNN and different nodes within each layer. For the purpose of clarity, we represented the MAS architecture module in Fig. 3.

The mathematical representations and notations used in this work are illustrated first for convenience. In a graph $G(V, E)$, with a set of nodes V and a set of edges E, each node i is associated with an original feature vector x_i and a true label y_i. Matrix $X = [x_1, x_2, \ldots, x_n]^T$ arranges all node vectors vertically. Let $A = [a_i, a_2, \ldots, a_n]^T$ represent the adjacency matrix. Node i has a collection of indices for its 1-hop neighbours stored in N_i. The GATE-GNN model is designed to predict the labels of nodes for which we have no information by learning a model $M = f(X, A)\mathbb{R}^{N\times 1}$ using the labels of a subset of nodes. At each message passing step l, $h_i^{(l-1)} \in \mathbb{R}^F$ represents the input features for node i, and its output vector is denoted as $h_i^{(l-1)} \in \mathbb{R}^{F'}$.

The low-order coefficient f_{ij} is calculated for an L-layer model to represent the relevance of features between nodes i and j, which are normalised among neighbouring nodes.

$$f_{ij} = \frac{\exp\left(\sigma\left(\alpha^T.\left[W^{(l)}h_i \parallel W^{(l)}h_j\right]\right)\right)}{\sum_{K\in N_i}\exp\left(\sigma\left(\alpha^T.\left[W^{(l)}h_i \parallel W^{(l)}h_j\right]\right)\right)}, \tag{2}$$

where $\parallel$ represents concatenation and σ is a non-linear activation function known as LeakyReLU, $W^{(l)} \in \mathbb{R}^{F'\times F}$ and $\alpha \in \mathbb{R}^{2F}$ are parameters adjusted during training. A higher value of f_{ij} indicates a stronger relationship between nodes i and j within their vicinity.

High-order coefficient generation involves utilising the neighbouring matrix $\hat{A} = A + I$ to represent topological information and creating high-order connections based on it, where I is an identity matrix. Initially, the link is weighted based on topological feature information.

$$h_v^{(l)} = \sigma\left(\sum_{u\in N(v)}\alpha_{uv}^{(l)}W^{(l)}h_u^{(l-1)} + \sum_{u\in N(v)}\sum_{w\in N(u)}\beta_{uvw}^{(l)}W^{(l)}h_w^{(l-1)}\right), \tag{3}$$

where $h_v^{(l)}$ is the node representation at layer l, $N(v)$ is the set of neighbours of node v, $\alpha_{uv}^{(l)}$ and $\beta_{uvw}^{(l)}$ are the first-order and second-order attention coefficients, respectively, $W^{(l)}$ is a weight matrix for layer l, and σ is an activation function. The first-order attention coefficient $\alpha_{uv}^{(l)}$ is computed using a self-attention mechanism. The second-order attention coefficient $\beta_{uvw}^{(l)}$ is computed using a soft-attention mechanism. The second-order attention coefficient measures the similarity or importance of node w for node v through the intermediate node u.

To generalise, higher-order attention is done by adding more terms that involve nodes that are further away from node v, such as $\sum_{u\in N(v)}$, $\sum_{w\in N(u)}$, $\sum_{z\in N(w)}$, $\gamma_{uvwz}^{(l)}W^{(l)}h_z^{(l-1)}$, where $\gamma_{uvwz}^{(l)}$ is the third-order attention coefficient.

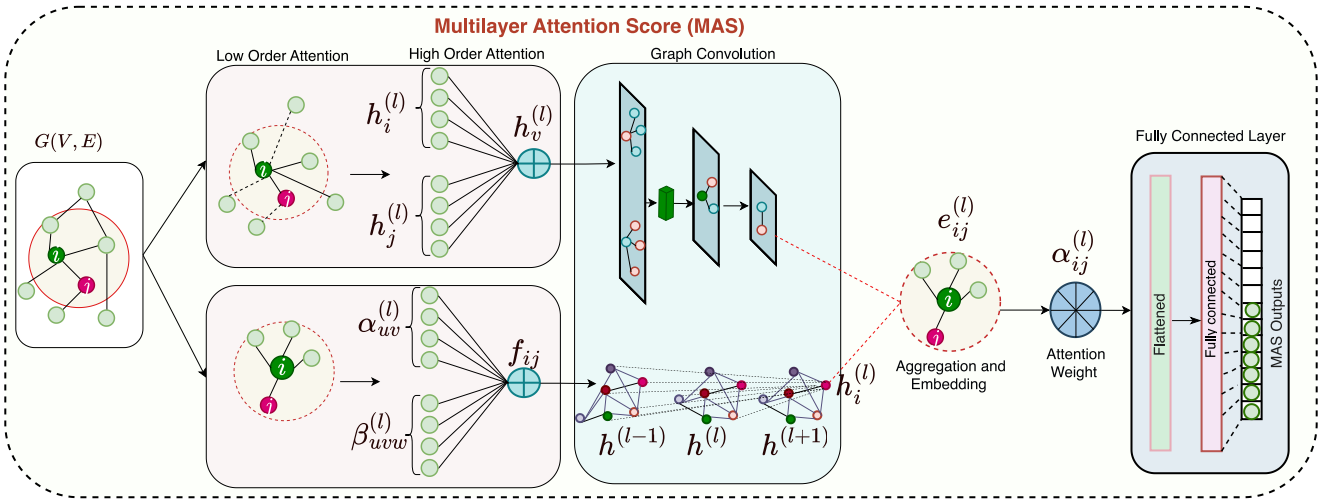


Fig. 3. Architecture of the multilayer attention score (MAS): where $G(V,E)$ is the input vector of nodes V and edges E, $h_i^{(l)}$ and $h_j^{(l)}$ are the feature vectors for the i and j nodes in each layer at low-level order attention, respectively, and $\alpha_{uv}^{(l)}$ and $\beta_{uvw}^{(l)}$ are the first and second higher-level order attention coefficients, respectively. $h_v^{(l)}$ and f_{ij} are the aggregated higher and low-level attention vectors, respectively. $h^{(l-1)}, h^{(l)}$, and $h^{(l+1)}$ are the topological node embedding layers; $h_i^{(l)}$ is the linear projection feature vector for each node and layer; $e_{ij}^{(l)}$ is the weight attention coefficient; and $\alpha_{ij}^{(l)}$ is the attention weight of the MAS module.

A linear projection matrix for each layer of the GNN transforms the node features first, producing a set of feature vectors for each node and each layer.

$$\mathbf{h}_i^{(l)} = \mathbf{W}^{(l)}\mathbf{x}_i, \quad \forall i \in \mathcal{V}, \forall l \in 1,\ldots,L, \tag{4}$$

where $\mathbf{x}_i$ is the input feature vector of the node. i, $\mathbf{h}_i^{(l)}$ is the feature vector of the node. i in layer l, $\mathbf{W}^{(l)}$ is the projection matrix for layer l, $\mathcal{V}$ is the set of nodes, and L is the number of layers.

To find the attention coefficients, a LeakyReLU activation function is applied to the dot product of the feature vectors of two nodes. This is followed by a softmax normalisation over all the nodes in each layer. This results in a set of attention matrices for each layer of the GNN, where each element represents the attention weight between two nodes in that layer.

$$e_{ij}^{(l)} = \sigma(\mathbf{a}^{(l)T}[\mathbf{h}_i^{(l)} \parallel \mathbf{h}_j^{(l)}]), \quad \forall_i, j \in \mathcal{V}, \forall_l \in 1,\ldots,L, \tag{5}$$

$$\alpha_{ij}^{(l)} = \frac{\exp(eij^{(l)})}{\sum_{k\in\mathcal{V}}\exp(e_{ik}^{(l)})}, \quad \forall_i, j \in \mathcal{V}, \forall_l \in 1,\ldots,L, \tag{6}$$

where $\mathbf{a}^{(l)}$ is the attention vector for layer l, $\parallel$ is the concatenation operator σ the LeakyReLU, and $\alpha_{ij}^{(l)}$ is the element of the attention matrix $\mathbf{A}^{(l)}$ for layer l.

Lastly, MAS is obtained by multiplying the attention matrices across the layers, resulting in a single attention matrix that captures the importance of each node's features across the layers. This matrix is used to aggregate the feature vectors of the nodes and their neighbours, weighted by the MAS for node representations at the GNFE module.

$$\mathbf{A_{MAS}} = \prod_{l=1}^{L}\mathbf{A}^{(l)}, \quad \mathbf{h_i} = \sum j \in \mathcal{V}\mathbf{A}_{ij}\mathbf{h}_j^{(L)}, \quad \forall_i \in \mathcal{V}, \tag{7}$$

where $\mathbf{A_{MAS}}$ is the multilayer attention matrix, and $\mathbf{h}_i$ is the final feature vector of node i.

4.3. Graph node feature extraction (GNFE)

The extraction functionality within our Graph Attention Ensemble GNN (GATE-GNN) is encapsulated by the Graph Node Feature Extraction (GNFE) module, which leverages the strengths of Graph Convolutional Networks (GCN). This method is tailored to elicit refined node representations by fusing intrinsic node attributes with the structural properties of the graph. The attention scores generated by the Multilayer Attention Score (MAS) module, enrich the nodes' feature sets. Our implementation of GCNs is specifically devised to unravel the complex non-linear dependencies encountered in graph network data.

The GNFE module is a Gumbel SoftMax trick to learning and select the most relevant features for each node based on its multiscale neighbourhood and fine-tune the GATE-GNN model on the graph data by employing information node feature extraction. Graph node feature extraction (GNFE) is a method that seeks to acquire significant and informative representations of the nodes in a graph by considering their attributes and structural interactions. Utilising this technique enhances the efficiency of GNNs for node classification tasks, which involve predicting the label or category of a node based on its attributes and the graph structure. The representation of the MAS architecture module is illustrated in Fig. 4.

4.3.1. Embedding process

GATE-GNN lets the model learn and process the low-dimensional vector representations (embeddings) for each node in the graph based on the features of the node and the structure of the graph. This is what it does during the embedding process, which is given as:

$$h_e^{(l)} = \sigma\left(W^{(l)}\sum_{u\in N(v)}\frac{1}{\sqrt{d_v d_u}}h_u^{(l-1)} + B^{(l)}h_v^{(l-1)}\right) \tag{8}$$

where $h_e^{(l)}$ is the node embedding at layer l, $N(v)$ is the set of neighbours of node v, d_v is the degree of node v, $W^{(l)}$ and $B^{(l)}$ are weight matrices for layer l.

The node embedding at layer l is a function of the node embedding at layer (l-1) and the embeddings of its neighbours at layer (l-1). A linear transformation $B^{(l)}$ changes the node embedding at layer (l-1). It is then added to the neighbours' embeddings, which are changed by another linear transformation, $W^{(l)}$. The aggregation of the neighbour embeddings is weighted by the inverse square root of the node degrees, which is a normalisation factor that prevents the embeddings from becoming too large or too small. The resulting embedding is passed through an activation function σ, which introduces non-linearity and allows the network to learn complex patterns.

4.3.2. How the GNFE works?

The primary purposes of employing graph node feature extraction (GNFE) methods in node classification, as identified in our study, are as follows: (1) it improves the ability of GNNs to express and generalise

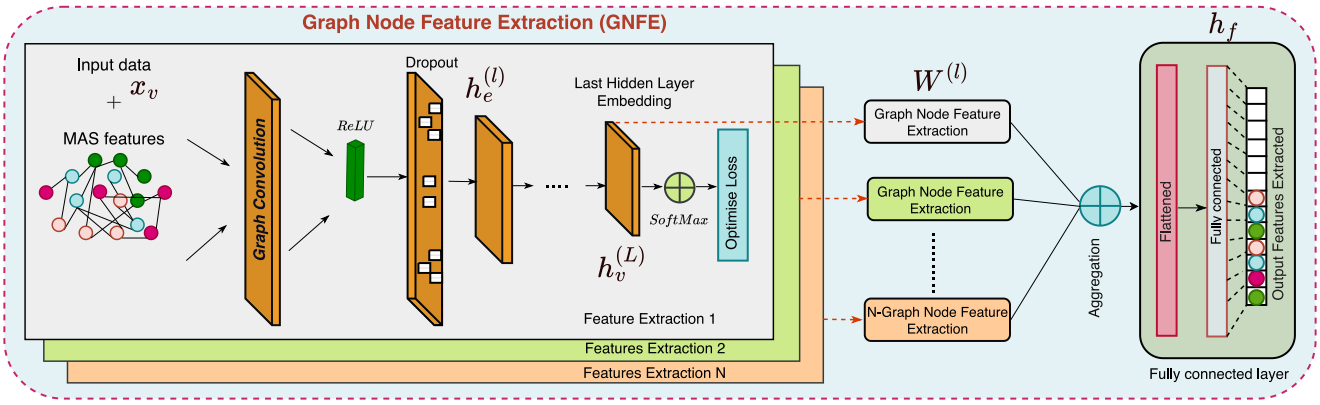


Fig. 4. Architecture of the graph node feature extraction (GNFE) process: where x_i is the input vector node, $h_e^{(l)}$ is the node embedding at layer l, $h_v^{(L)}$ is the final hidden embedding layer, $W^{(l)}$ is the aggregated weight at layer l, and h_f is the final feature obtained from the extracted process.

by capturing the semantic and contextual information of the nodes, including both global and local patterns of the graph; (2) it reduces the dimensionality and sparsity of the node features, which helps to reduce the computational and memory requirements of GNNs, particularly for large-scale graphs; (3) it also allows for the transferability and interpretability of the node features, making it easier to apply GNNs to different domains and tasks; (4) it provides valuable insights into the node features and the structure of the graph.

The GNFE consists of multiple layers of graph convolution operations, which aggregate the features of a node and its neighbours, and apply a non-linear transformation. The GNFE is represented as follows:

$$h_v^{(0)} = x_v, \tag{9}$$

$$h_v^{(l)} = \sigma\left(\sum_{u \in N(v) \cup v} \frac{1}{\sqrt{d_v d_u}} W^{(l)} h_u^{(l-1)}\right), \tag{10}$$

$$h_f = h_v^{(L)}, \tag{11}$$

where $h_v^{(l)}$ is the node representation at layer l, x_v is the input feature vector of node, $W^{(l)}$ is a weight matrix for layer l, and L is the final number of layers. The final node representation h_v is used for downstream tasks for the node classification task and h_f is the final feature extracted from GNFE module.

4.4. Node ensemble and transfer learning (NETL)

In the domain of deep learning, the computational expense of model training is significant. To mitigate this, we introduce the Node Ensemble and Transfer Learning (NETL) technique in tandem with a transfer learning strategy to enhance our proposed model's efficacy. The NETL method accentuates critical elements within the node graph, facilitating the transfer of learned parameters from existing classifiers to new ones. This approach leverages the densely connected layers of the MAS, where each node interlinks with every other node in successive layers—a feature the NETL optimally exploits.

In order to use transfer learning, we utilise a domain-adversarial training technique that aligns the distributions of the node representations in both the source and target graphs. This alignment is done while ensuring that the discriminative information for the target task is preserved. We propose an adversarial classifier that aims to differentiate between the node representations of the source and target graphs (Fig. 2 C). Additionally, we incorporate a domain confusion loss to deceive the classifier by ensuring that the node representations become indistinguishable. The domain-adversarial training technique is expressed as:

$$\min_{\theta} \max_{\phi} \mathbb{E}v \sim V_s[\log D\phi(h_v^{\theta})], \tag{12}$$

$$\mathbb{E}v \sim V_t[\log(1 - D\phi(h_v^{\theta}))] - \lambda \mathcal{L}(h_v^{\theta}, y_v), \tag{13}$$

By adding Eqs. (12) and (13), we obtained the output of NETL as:

$$h_t = \min_{\theta} \max_{\phi} \mathbb{E}v \sim V_s + \mathbb{E}v, \tag{14}$$

where θ and ϕ are the parameters of the NETL and the adversarial classifier, respectively, D_ϕ is the adversarial classifier, h_t is the NETL output, h_v^{θ} is the node representation produced by the NETL, λ is a trade-off parameter, and $\mathcal{L}$ is the loss function for the target task.

Nevertheless, the increasing number of parameters and the corresponding weights to be learned pose a challenge during the training phase, a task the GNFE module alone struggles to manage efficiently. The feature sets refined by the GNFE equip the NETL to discern temporal dependencies across nodes, thereby bolstering the model's predictive capabilities. The NETL module is adept at assimilating diverse feature types within the graph traffic network, encompassing both numerical and categorical data.

Post the embedding phase and reshuffling of graph data, two primary processes are initiated: the construction of self-attention feature maps and the formation of an attention map. The self-attention feature maps empower the NETL to discern inter-node relationships within the input sequence. By allocating a relevance score to each node segment, the NETL synthesises these weights with the input features to yield an output that encapsulates the temporal nodes' contextual significance. Consequently, this mechanism equips the network to manage long-range dependencies and inputs of variable lengths effectively.

4.5. Graph ensemble weight attention (GEWA)

The Graph Ensemble Weight Attention (GEWA) constitutes the core of our proposed Graph Attention Ensemble GNN (GATE-GNN). For a given input graph let $G = (V, E)$ be a graph with n nodes and m edges, where each node $v \in V$ has a feature vector $x_v \in \mathbb{R}^d$. We want to learn a node representation $h_v \in \mathbb{R}^k$ that captures the structural and semantic information of the graph. The GEWA framework integrates features extracted from the Graph Node Feature Extraction (GNFE) and the Graph Ensemble and Transfer Learning (NETL). These features are further processed using GEWA-Boosting. Employing spatial graph attention, the GEWA module aggregates the features of neighbouring nodes with varying attention weights to encapsulate spatial correlations and dependencies within the graph data networks.

The GEWA framework utilises a graph convolution network to encode the interdependencies among nodes, dynamically assigning attention weights to capture the nodes' interactions and the graph's heterogeneity. In addressing the challenges of bias and variance in spatial processes, the GEWA aims to discern the non-linear relationships

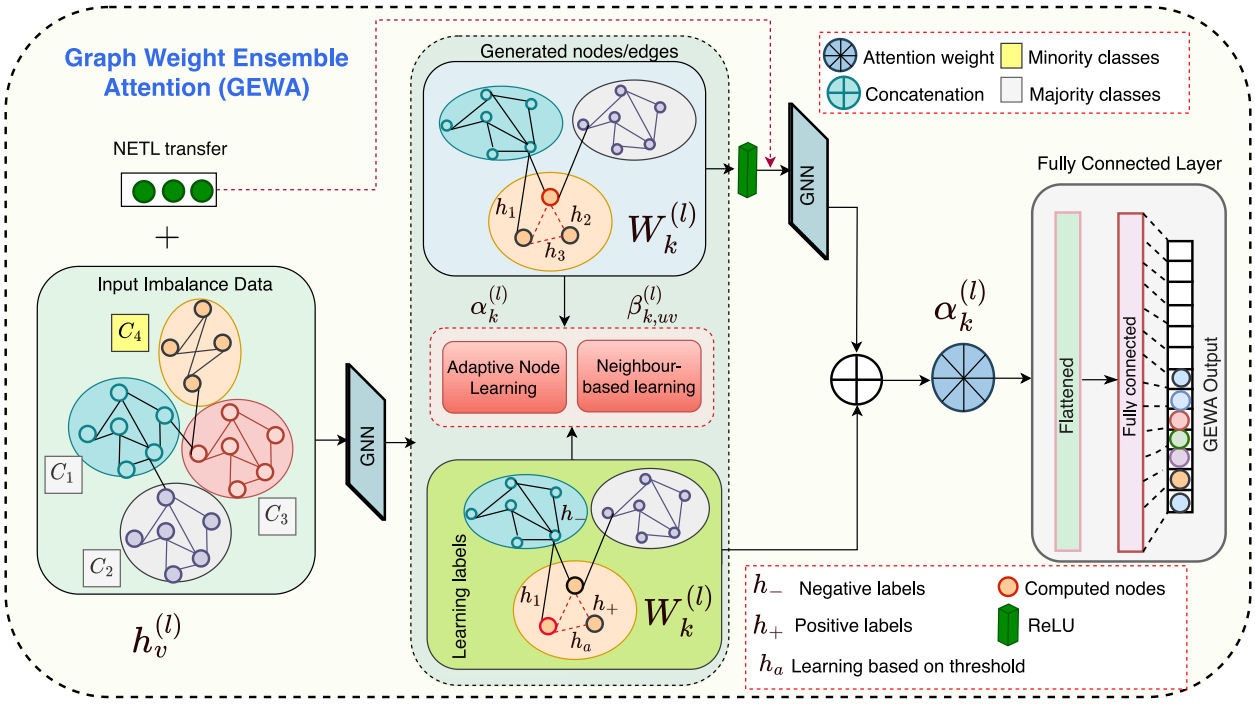


Fig. 5. The architecture of Graph Weight Ensemble Attention (GEWA): where $h_v^{(l)}$ is the node embedding or representation at layer l, $W_k^{(l)}$ is the weight matrix for the kth attention head at layer l, $\alpha_k^{(l)}$ is the GEWA mechanism's aggregated weight, $\beta_{(uv)}^{(l)}$ is the attention coefficient, h_{-+} are the positive and negative labels, respectively, and h_a are the learning based on the weight threshold.

between geographic elements for the classification of weight attention nodes (Fig. 5).

To do this, we use a GEWA with L layers, where each layer consists of an ensemble of K attention heads (8 heads). Each attention head computes a weighted average of the node features and the features of its neighbours using a learnable attention mechanism. The output of each layer is the concatenation of the outputs of all the attention heads. Ideally, the GEWA can be written as:

$$h_v^{(0)} = x_v, \tag{15}$$

$$h_v^{(l)} = \Vert \left(\left[\alpha_k^{(l)} \sum_{u \in N(v)} \beta_{k,uv}^{(l)} W_k^{(l)} h_u^{(l-1)} \right]_{k=1}^{K} \right), \tag{16}$$

$$h_v = h_v^{(L)}, \tag{17}$$

where $h_v^{(l)}$ is the node representation at layer l, $N(v)$ is the set of neighbours of node v, $\alpha_k^{(l)}$ is a scalar weight for the kth attention head at layer l, $\beta_{k,uv}^{(l)}$ is the attention coefficient for the edge (u, v) at layer l, $W_k^{(l)}$ is a weight matrix for the kth attention head at layer l, and $\Vert$ is the concatenation operation.

The attention coefficients $\beta_{k,uv}^{(l)}$ are computed using a softmax function over the dot product of the node features, scaled by a factor $\sqrt{d_k}$, where d_k is the dimension of the node features after applying $W_k^{(l)}$. The attention coefficients can be written as:

$$\beta_{k,uv}^{(l)} = \frac{\exp\left(\frac{(W_k^{(l)} h_u^{(l-1)})^T (W_k^{(l)} h_v^{(l-1)})}{\sqrt{d_k}} \right)}{\sum_{w \in N(v)} \exp\left(\frac{(W_k^{(l)} h_w^{(l-1)})^T (W_k^{(l)} h_v^{(l-1)})}{\sqrt{d_k}} \right)}. \tag{18}$$

The scalar weights $\alpha_k^{(l)}$ are learned by minimising a loss function that measures the discrepancy between the node representations and some target values. For example, given by the task of our problem is node classification, the target values is the node labels, and the loss function the cross-entropy loss. To do this, the scalar weights can be written as:

$$\alpha_k^{(l)} = \arg\min_{\alpha} \mathcal{L} \left(\Vert \left(\left[\alpha \sum_{u \in N(v)} \beta_{k,uv}^{(l)} W_k^{(l)} h_u^{(l-1)} \right]_{k=1}^{K} \right), y_v \right), \tag{19}$$

where $\mathcal{L}$ is the loss function and y_v is the target value for node v. To perform parameter sensitivity analysis, we vary the values of the model parameters (such as $W_k^{(l)}$ and $\alpha_k^{(l)}$) and observe how the node representations and the model performance change. In this case, since the local sensitivity information is also relevant, we employed the global sensitivity information to actually discern the information by incrementing and observing the resulting impact on node representations, the variance of the node representation, and by randomly selecting the obtained parameter values (λ_1 and λ_2) and assessing the model performance using the imbalance ratio (IR) values.

4.6. Classifier

The Classifier module of our Graph Attention Ensemble GNN (GATE-GNN) system capitalises on the features extracted by the Graph Ensemble Weight Attention (GEWA) module. It integrates an attention map, global attention, and a multi-headed policy to execute the final classification task. The global attention mechanism computes scores for each feature segment obtained from GEWA, utilising the entire sequence of the attention map. This process aids in understanding the holistic importance and structure of graph data within the graph network. However, processing the full input for each component can lead to computational inefficiency. To mitigate this, we incorporate ensemble and transfer learning methods.

In contrast, the multi-headed attention mechanism calculates scores for each graph data component, drawing from various subsets or projections of the graph network input sequence. This multi-headed policy

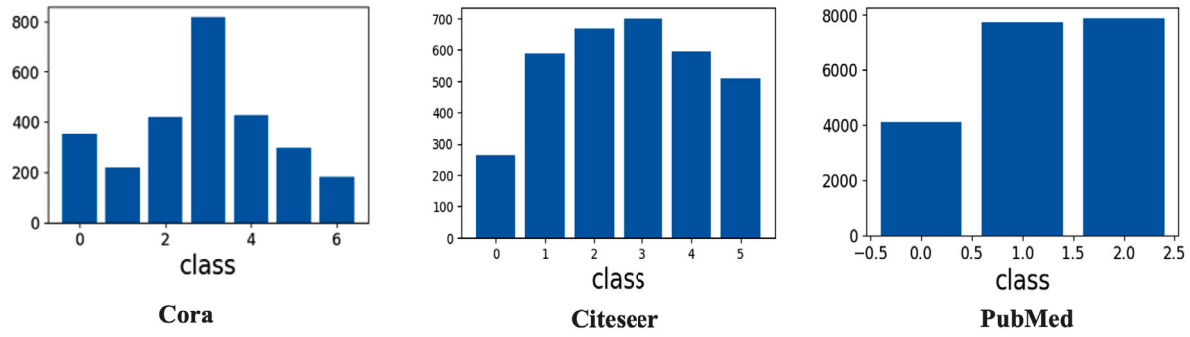


Fig. 6. Distribution of class numbers in the Cora, Citeseer, and PubMed datasets.

evaluates several scores for each node by considering different combinations or features of the network nodes, allowing the model to capture diverse representations and perspectives of the input data. The final classification is derived by concatenating the outputs from the global attention policy and attention map, leading to a comprehensive node classification.

The classifier module, illustrated in Fig. 2E, is designed to be computationally efficient. By leveraging the significant features already extracted from the GEWA module, we circumvent the need to compute the intricate dependencies across the entire graph map. The integration of ensemble and transfer learning techniques not only enhances the model's performance but also conserves computational resources.

We assumed let y_v be the class label of node v in the target graph $G_t = (V_t, E_t)$, where each node $v \in V_t$ has a feature vector $x_v \in \mathbb{R}^d$. We want to learn a classifier $f : \mathbb{R}^k \to C$, where C is the set of classes and k is the dimension of the node representation $h_v \in \mathbb{R}^k$ produced by the GEWA framework.

To do this, we use a softmax function that computes the probability of node v belonging to class $c \in C$ based on the node representation h_v. The classifier is represented as:

$$f(h_v) = \text{softmax}\left(W_c h_v + b_c\right), \tag{20}$$

where $W_c \in \mathbb{R}^{k\times|C|}$ and $b_c \in \mathbb{R}^{|C|}$ are the weight matrix and the bias vector for the classifier, respectively.

The classifier is trained by minimising the cross-entropy loss between the predicted probabilities and the true labels, using the node representations from the GATE-GNN model as inputs. The loss function of proposed GATE-GNN is given as:

$$\mathcal{L} = -\sum_{v\in V_t}\sum_{c\in C} y_{vc} \log f(h_v)_c, \tag{21}$$

where y_{vc} is the indicator function that equals 1 if node v belongs to class c, and 0 otherwise, and $f(h_v)_c$ is the probability of node v belonging to class c according to the classifier.

In general, the objective function of the proposed GATE-GNN model is represented as:

$$C_{GATE} = A_{MAS} + h_f + \lambda_1(h_t) + \lambda_2(h_v + \alpha_k^{(l)}), \tag{22}$$

where λ is the hyperparameter of the object function of the proposed GATE-GNN model and $\alpha_k^{(l)}$ is the aggregated score obtained by GEWA module.

5. Experimental results

This section delineates the experimental results of our study, systematically divided into ten subsections for clarity and depth of analysis. Section 5.1 details the datasets utilised in our experiments, laying the foundation for the subsequent analyses. Section 5.2 describes the methodology employed to generate imbalanced data, a critical aspect of our research. Section 5.3 elucidates key experimental hyperparameters, providing insights into the setup and configurations of our experiments. Section 5.4 introduces the methods we compared our approach against, establishing a benchmark for evaluation. Sections 5.5 to 5.10 are dedicated to presenting the experiment results, each focusing on a distinct aspect: Section 5.5 addresses accuracy, Section 5.6 examines the impact of skewness on results, Section 5.7 explores resilience to feature perturbation, Section 5.8 assesses training time, Section 5.9 examines the imbalance ratio (IR), and Section 5.10 detailed the analysis of parameter sensitivity. Together, these subsections comprehensively cover the various dimensions of our experimental inquiry.

Table 1
Selected datasets for the experiments.

Dataset	Cora	NELL	Citeseer	PubMed
Nodes	2708	154,213	3327	19,717
Edges	10,556	266,144	9104	88,648
Classes	7	210	6	3
Features	1433	5414	3703	5414

5.1. Preprocessing and description of datasets

Our experimental analysis utilises four distinct datasets, three from citation networks and one from the Never-Ending Language Learning (NELL) project.

- **Citation networks**: These datasets represent academic papers as nodes, with edges indicating citation links. The labels correspond to the journals of publication. Each node features a one-hot vector for article keywords. The classification task involves categorising unlabelled articles into specific domains using labelled nodes and hyperlinks. We employ three well-known datasets – Cora, Citeseer, and Pubmed – which are benchmarks in graph-based studies (Kipf & Welling, 2016; Sen et al., 2008).
- **Never-ending language learning (NELL)**: Comprising a substantial bipartite graph with 210 unique node types, the NELL dataset represents relationships as triplets (e_1, r, e_2). For each triplet, two edges are added to the graph, leading to a binary, symmetric adjacency matrix. Nodes are described by sparse feature vectors of 5414 dimensions (Table 1).

Preprocessing includes ensuring no nodes have connecting edges, the absence of self-loops, and non-directionality in network edges. This process effectively doubles the edge count due to bidirectional information. We analysed degree distribution and inter-class edge connections, crucial for understanding graph structure and homophily. The class distribution in the Cora, Citeseer, and PubMed datasets is illustrated in Fig. 6.

To provide a direct visual understanding of the datasets, we present visualisations for Cora, Citeseer, and PubMed in Fig. 7, where nodes and edges are colour-coded according to their classes.

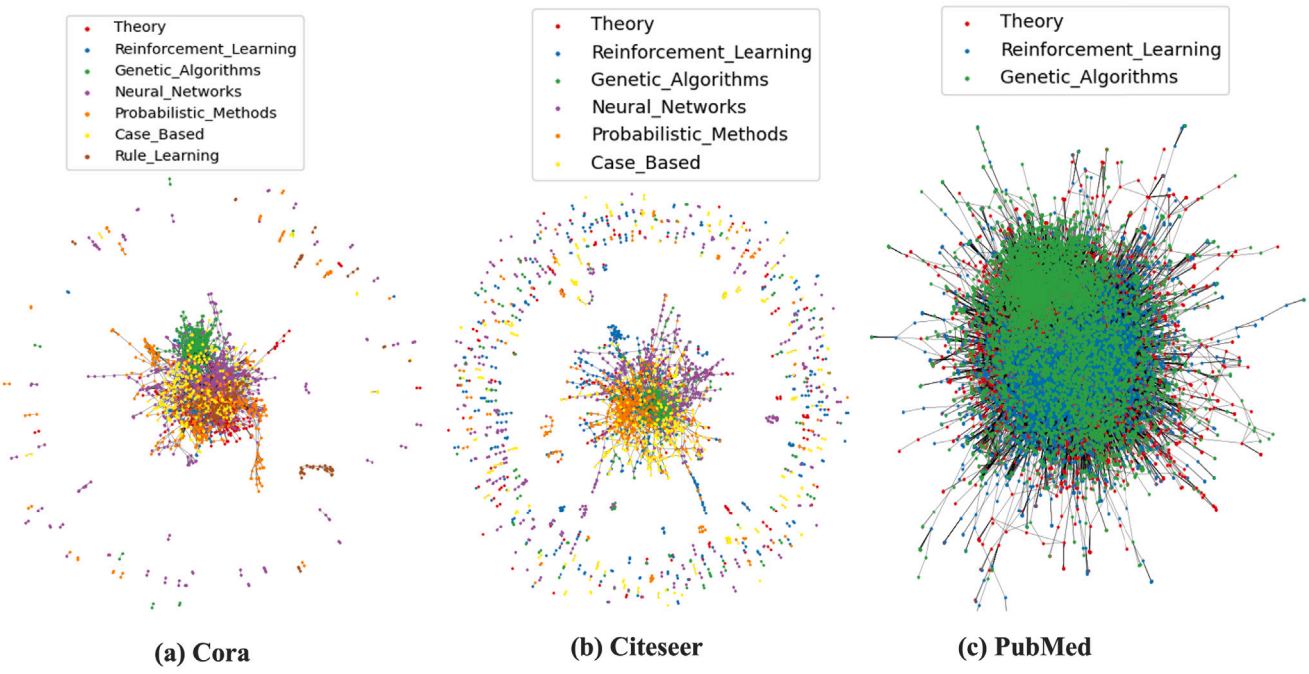


Fig. 7. Visualisation of the Cora, Citeseer, and PubMed datasets, with nodes and edges colour-coded to represent different classes.

The comprehensive analysis of these datasets is crucial for validating the effectiveness of our GATE-GNN model. Although the NELL dataset's complexity and diverse classifier categories precluded its visualisation, our exploration focuses in-depth on the citation network datasets.

5.2. Simulated imbalanced datasets

Simulated imbalanced datasets are crucial for examining the performance of GNNs in scenarios where node classification is imbalanced, with some classes having significantly fewer instances than others. To create these datasets, we applied the Pareto Principle (80–20 rule) and randomly designated one class as the majority, while the rest were treated as minority classes. Following the approach in Kipf and Welling (2016) and we selected 30 samples from each class to form the training set. Considering the imbalance dataset represented by the equation:

$$N_i = \begin{cases} 30 & \text{if } i = C, \\ s & \text{otherwise,} \end{cases} \tag{23}$$

where N_i denotes the number of samples in class i, C represents the majority class, and s signifies the number of samples in each minority class. We can adjust the imbalance level of the training dataset by varying s. For instance, in the Cora dataset with 7 distinct classes, we keep one class at 30 samples and alter the sample size for the remaining 6 classes. The training set is constructed by randomly choosing a specific number of samples in each iteration and the test set follows the partitioning method from Kipf and Welling (2016).

Node dropping is employed to simulate the imbalanced datasets. For a graph G, node dropping involves randomly removing vertices and their associated edges, adjusting the node types to meet imbalance requirements. The probability of dropping a node is uniform across the graph, that maintains the average node degree.

5.3. Parameter settings

The GATE-GNN framework encompasses three primary classifiers: GCN, GraphSAGE, and GAT, corresponding to the GATE-GCN, GATE-GraphSAGE, and GATE-GAT models, respectively. Each model undergoes training for up to 200 epochs using the Adam optimiser. For the Cora, Citeseer, and PubMed datasets, we set the hidden unit size to 16 with an $L2$ regularisation value of 5e-4. For the NELL model, the number of hidden units is increased to 128, and $L2$ regularisation is applied with a coefficient of 1e-5. The dropout rate is set to 0.5, weight decay is 5e-4, weight loss (λ) is 1e-6, and imbalance ratio set is set at 0.2 to 0.6.

The hyperparameter configurations for the GATE-GNN models are as follows:

- **GATE-GCN**: The activation function is the Rectified Linear Unit (ReLU). Learning rates for Cora, Citeseer, PubMed, and NELL are set to 0.01, 0.01, 0.01, and 0.005, respectively.
- **GATE-GraphSAGE**: Also employs ReLU as the activation function. The first layer samples 25 neighbours, while the second layer samples 10. The learning rates for Cora, Citeseer, PubMed, and NELL are 0.001, 0.001, 0.0005, and 0.0001, respectively.
- **GATE-GAT**: Utilises the Exponential Linear Unit (ELU) in the first layer and the softmax function (σ) in the second layer. It employs 8 attention heads ($K = 8$). Learning rates are set to 0.001 for Cora, Citeseer, and PubMed, and 0.0005 for NELL.

The evaluation focuses on the single experiment yielding the highest accuracy. The final outcome is the average accuracy across 10 runs, each initiated with different random sample splits. Distinct random seeds are used for node removal in each iteration, but the 10 seeds are consistent across all models. Experiments were conducted on the Google Compute Engine Backend, equipped with a GPU and 12.68 GB of RAM.

5.4. Baseline methods

To evaluate the effectiveness of our proposed GATE-GNN model, we compare it with a range of baseline methods, categorised into five groups as outlined in Table 2.

- **Standard GNN models:**
  - *GCN*: The Graph Convolutional Network, introduced by Kipf and Welling (2016), employs a first-order approximation of Chebyshev polynomials for constructing node embeddings.

Table 2
Percentage summary of results in terms of classification accuracy.

Model	Cora	NELL	Citeseer	PubMed
GCN	65.6 ± 0.8	68.5 ± 1.4	62.2 ± 0.5	71.8 ± 0.6
GraphSAGE	66.3 ± 0.8	69.6 ± 1.3	59.7 ± 0.6	69.7 ± 0.6
GAT	67.4 ± 0.7	70.3 ± 1.6	60.3 ± 0.6	66.2 ± 0.7
N-GCN	67.3 ± 0.6	73.3 ± 1.2	65.4 ± 0.3	72.3 ± 0.3
SGC	69.7 ± 0.8	67.1 ± 1.4	59.4 ± 0.5	66.9 ± 0.5
GCN-FL	67.8 ± 1.2	71.2 ± 1.2	65.1 ± 0.8	72.4 ± 0.8
GraphSAGE-FL	66.5 ± 1.2	72.1 ± 1.1	59.5 ± 0.8	69.7 ± 1.3
GAT-FL	67.4 ± 1.3	72.6 ± 1.0	61.3 ± 0.7	69.2 ± 1.2
GCN-CB	70.6 ± 0.9	72.9 ± 1.4	65.1 ± 0.6	72.3 ± 0.8
GraphSAGE-CB	66.3 ± 0.9	69.8 ± 1.4	59.7 ± 0.9	70.1 ± 0.9
GAT-CB	67.4 ± 1.0	73.4 ± 1.5	60.3 ± 0.7	69.3 ± 0.9
GCN-RS	70.4 ± 1.0	68.9 ± 2.1	61.8 ± 1.1	70.4 ± 1.0
Boosting-GCN	73.2 ± 0.7	74.9 ± 1.0	65.7 ± 0.7	73.1 ± 0.7
Boosting-GraphSAGE	72.4 ± 1.0	75.3 ± 1.2	63.4 ± 1.0	70.4 ± 1.1
Boosting-GAT	73.5 ± 0.5	75.5 ± 1.0	64.3 ± 0.8	69.7 ± 0.7
PMLP-GCN	75.6 ± 0.9	77.4 ± 1.4	68.5 ± 0.6	76.5 ± 0.5
PMLP-SGC	75.3 ± 1.0	79.5 ± 0.6	67.8 ± 0.6	76.0 ± 1.4
PMLP-APP	80.2 ± 1.4	78.5 ± 1.5	72.5 ± 1.4	79.8 ± 1.9
INS-GNN_{lp}	75.2 ± 0.9	78.4 ± 1.1	66.2 ± 0.6	76.2 ± 0.5
INS-GNN_{ica}	78.4 ± 0.6	78.8 ± 0.9	68.2 ± 1.5	77.3 ± 1.1
GNN-CL	74.5 ± 0.5	70.5 ± 0.6	63.8 ± 0.3	76.5 ± 0.5
GNN-CL_O	70.1 ± 0.8	72.1 ± 0.6	62.5 ± 0.6	65.6 ± 0.9
GNN-CL_M	72.6 ± 0.5	73.4 ± 0.8	63.7 ± 1.0	67.4 ± 1.2
GNN-CL_C	71.3 ± 1.0	73.6 ± 1.2	62.7 ± 0.9	63.6 ± 0.9
GATE-GCN	81.5 ± 0.8	82.5 ± 1.1	71.1 ± 0.7	**82.3 ± 0.7**
GATE-GraphSAGE	79.4 ± 1.1	82.1 ± 1.0	70.5 ± 1.0	78.5 ± 1.1
GATE-GAT	**83.5 ± 1.1**	**85.5 ± 1.0**	**75.5 ± 0.8**	77.7 ± 0.8

NB: The models' top performance is indicated in bold.

  - *GraphSAGE*: As described by Hamilton et al. (2017), this model aggregates vertex features in the vicinity of a target vertex to create embeddings.
  - *GAT*: The Graph Attention Network was proposed by Velickovic et al. (2017) that utilises an attention mechanism for embedding computation.
  - *SGC*: Simplifying Graph Convolutional Network, introduced by Wu et al. (2019), streamlines the GCN model by removing non-linearities.
  - *N-GCN*: This approach, by Abu-El-Haija, Kapoor, Perozzi, and Lee (2020), involves convolutions at various neighbourhood sizes and aggregates their outcomes.

- **Focused loss function models:**
  - *GCN-FL, GraphSAGE-FL, GAT-FL*: These models adapt GCN, GraphSAGE, and GAT respectively, with a focal loss function inspired by Lin et al. (2017) to enhance performance on imbalanced datasets.

- **Class-balanced models:**
  - *GCN-CB, GraphSAGE-CB, GAT-CB*: These models, as proposed by Li and Nabavi (2023), balance the loss function according to class distribution, employing a sampling technique for equitable node selection.
  - *PMLP-GCN, PMLP-SGC, and PMLP-APP*: introduced an intermediate model classed as Propagation MLP (PMLP) that adopts GNN's architecture in testing to achieve better performance for node classification tasks (Yang et al., 2023), where SGC is simplify graph convolution and APP is approximate personalised propagation.
  - *INS-GNN*: Juan et al. (2023) utilises a self-supervised learning method to address the problem of graph imbalance learning, where some classes of nodes are underrepresented in the graph. Sampling a balanced subgraph from the original graph using an imbalanced node sampling strategy that preserves the graph structure and the class distribution and using a self-supervised contrastive loss that encourages the node representations to be similar within the same class and dissimilar across different classes. The INS-GNN has two variants: INS-GNN_{lp} and INS-GNN_{ica}, where lp is the label propagation and ica is the iterative classification algorithm.
  - *GNN-CL*: Li et al. (2024) implemented a graph-based oversampling technique that generates new nodes and edges for the minority classes based on the smoothness and homophily principles of the graph. It combines a graph classification loss and a metric learning loss and adjusts the weights of the losses dynamically based on the curriculum learning strategy. Additionally, GNN-CL has three variants: GNN-CL_O, GNN-CL_M, and GNN-CL_C, where CL_O is the curriculum learning oversampling strategy, CL_M curriculum learning metric loss from the model, and CL_C is the curriculum learning mechanism.

- **Resampling model:**
  - *GCN-RS*: This model, detailed by Wang, Chen, Li, and Wang (2021), addresses noise and scalability issues in large-scale graphs, employing a scalable training method using subgraphs.

- **Boosting models:**
  - *Boosting-GCN, Boosting-GraphSAGE, Boosting-GAT*: Shi et al. (2021) introduced these models to leverage the Sylvie training framework and combining one-bit quantisation with GNNs for effective learning.

Each group represents a distinct approach to graph neural network design and optimisation. This offers a comprehensive perspective on the current state-of-the-art techniques in this domain. The comparison with these varied models allows us to thoroughly assess the performance and advantages of GATE-GNN in handling imbalanced node classification challenges.

5.5. Classification accuracy

The performance of the GATE-GNN model was evaluated against a comprehensive set of baseline methods as shown in Table 2. Our implementation was executed on both Keras and Tensorflow frameworks to leverage their compatibility for seamless model integration.

Node dropping, as described in Section 5.2, was employed to simulate imbalanced test data, with the imbalance parameter s set to 3, as per Eq. (23). The training and validation sets followed the methodology of Kipf and Welling (2016), using 500 nodes for validation and 1000 nodes for testing, to maintain consistency with established benchmarks.

Table 2 presents the classification accuracy of various models, highlighting GATE-GNN's superior performance over traditional GNNs and contemporary approaches in handling imbalanced datasets. Notably, GATE-GNN outperforms alternative methods by an average margin of 5%–10%.

The INS-GNN, PMLP, GNN-CL, and Boosting-GNN models, employing multi-scale convolution operations, showed modest improvements in performance compared to standard GCN. The PMLP and INS-GNN models showcase effectiveness when compared to the traditional models. However, our approach's integration of weight attention ensemble and transfer-learning methods has led to significant performance enhancements. Resampling and reweighting methods showed some improvement in accuracy, but not as pronounced as expected. Particularly, Boosting-GNN did not effectively manage the bias–variance trade-off in integrating multiple base GNN classifiers.

Our GATE-GNN model demonstrates a significant increase in classification accuracy on the Cora, NELL, Citeseer, and PubMed datasets, with average improvements of 4.5% for Cora and NELL, 8.6% for Citeseer, and 5.2% for PubMed when compared to INS-GNN and PMLP models.

Table 3
The GATE-GNN results with different number of classifiers in terms of accuracy (in %).

Number of base classifier		100 epochs			200 epochs		
Model	Classifier	Cora	Citeseer	PubMed	Cora	Citeseer	PubMed
Boosting-GNN	3	75.7 ± 2.4	65.5 ± 2.5	63.9 ± 2.4	75.4 ± 2.1	65.6 ± 1.1	72.0 ± 0.8
PMLP		76.5 ± 1.9	68.2 ± 2.1	65.9 ± 2.4	75.8 ± 0.7	68.8 ± 0.6	75.9 ± 0.6
INS-GNN		74.8 ± 1.3	66.5 ± 1.3	**75.9 ± 1.4**	76.4 ± 1.3	68.4 ± 1.1	74.6 ± 0.9
GNN-CL		65.9 ± 0.5	62.5 ± 0.7	74.2 ± 1.1	72.5 ± 1.1	65.2 ± 1.1	61.4 ± 0.9
GATE-GNN		**79.6 ± 1.4**	**69.6 ± 1.1**	75.7 ± 0.9	**84.5 ± 1.4**	**75.5 ± 1.1**	**81.5 ± 0.8**
Boosting-GNN	5	73.2 ± 0.7	65.7 ± 0.7	73.1 ± 0.7	75.6 ± 2.3	65.9 ± 0.5	73.1 ± 1.1
PMLP		75.8 ± 0.9	**67.4 ± 0.9**	68.5 ± 0.9	75.8 ± 0.7	67.7 ± 0.9	76.5 ± 0.9
INS-GNN		75.5 ± 1.9	**67.4 ± 1.5**	**75.6 ± 1.3**	78.5 ± 1.2	68.5 ± 1.1	75.7 ± 1.3
GNN-CL		66.5 ± 0.9	62.2 ± 0.8	73.5 ± 0.6	73.5 ± 1.1	63.2 ± 1.1	72.4 ± 0.9
GATE-GNN		**78.3 ± 1.4**	66.7 ± 1.7	74.9 ± 0.8	**83.3 ± 1.4**	**72.1 ± 1.1**	**77.8 ± 0.8**
Boosting-GNN	7	73.5 ± 1.4	64.5 ± 0.5	73.5 ± 1.4	74.1 ± 2.7	64.7 ± 0.4	73.5 ± 0.8
PMLP		74.4 ± 1.5	65.5 ± 1.5	66.7 ± 1.5	75.3 ± 1.3	**71.4 ± 1.3**	72.7 ± 1.3
INS-GNN		76.6 ± 1.3	**68.4 ± 0.9**	**75.5 ± 1.2**	79.5 ± 1.5	68.4 ± 1.3	**78.4 ± 1.0**
GNN-CL		70.2 ± 0.7	62.4 ± 0.9	72.3 ± 1.1	72.6 ± 1.1	62.5 ± 0.9	70.2 ± 0.9
GATE-GNN		**79.3 ± 0.9**	67.9 ± 0.8	74.4 ± 0.9	**81.9 ± 0.9**	71.1 ± 0.8	77.7 ± 0.8
Boosting-GNN	9	72.0 ± 0.5	63.6 ± 0.5	72.0 ± 0.5	73.9 ± 2.0	64.2 ± 0.3	72.6 ± 1.1
PMLP		75.5 ± 0.6	63.6 ± 0.9	67.9 ± 0.9	76.2 ± 1.4	67.4 ± 1.4	**77.5 ± 1.5**
INS-GNN		74.9 ± 0.9	**67.6 ± 0.9**	74.4 ± 1.1	78.9 ± 1.5	66.8 ± 1.2	75.8 ± 1.4
GNN-CL		68.5 ± 1.1	63.5 ± 1.2	68.9 ± 1.1	73.1 ± 1.1	62.5 ± 1.1	72.4 ± 0.9
GATE-GNN		**77.9 ± 0.6**	65.7 ± 0.6	**74.7 ± 0.9**	**81.2 ± 1.1**	**70.6 ± 0.6**	77.3 ± 0.8
Boosting-GNN	11	73.0 ± 0.7	64.5 ± 0.6	73.0 ± 2.3	74.1 ± 2.3	65.1 ± 0.3	71.5 ± 0.7
PMLP		74.7 ± 1.5	66.5 ± 1.5	68.2 ± 1.5	79.3 ± 1.1	66.5 ± 0.6	73.8 ± 0.6
INS-GNN		76.5 ± 1.3	**66.9 ± 1.5**	74.2 ± 1.5	75.6 ± 1.3	68.5 ± 0.9	75.4 ± 1.3
GNN-CL		67.7 ± 1.5	64.4 ± 1.3	68.4 ± 1.1	72.2 ± 1.1	63.4 ± 1.1	72.4 ± 1.1
GATE-GNN		**77.8 ± 1.1**	65.3 ± 0.6	**74.3 ± 0.9**	**82.1 ± 1.1**	**69.7 ± 0.6**	**76.8 ± 0.8**

NB: The models' top performance is indicated in bold.

5.6. Evaluating classification accuracy across varying classifier quantities and epochs

In assessing the performance of imbalanced datasets, we evaluated the impact of varying the number of base classifiers. Table 3 presents the results for the GATE-GNN model based on GCN, trained over different epochs (100 and 200) and with varying quantities of classifiers (3, 5, 7, 9, 11). The purpose was to observe the model's performance across these variables and compare it with the established GNNs model.

Our findings indicate that GATE-GNN consistently outperforms baseline models across different training epochs and classifier quantities. For example, the average accuracy gains were 5.1%, 2.3%, and 3.7% for the three different classifier quantities. These gains were even bigger after 200 epochs, reaching 8.2%, 6.5%, and 5.1%, respectively. This demonstrates the effectiveness and precision of our model, particularly as the number of base classifiers changes. However, after 100 trials, the PMLP and INS-GNN models do better than the proposed GATE-GNN model by 0.5% to 1% on classifiers 5, 7, and 9 on the Citeseer dataset. After 200 trials, classifiers 7 and 9 on the PubMed dataset also do better.

Our model's aggregation technique, GATE-GNN, contributes to its enhanced efficiency. Notably, we found that the accuracy correlates with the number of base classifiers. However, as the quantity of base classifiers increases, a diminishing return in accuracy is observed, suggesting an optimal balance in the number of classifiers used. This phenomenon was examined by maintaining the sequence of classes and randomly selecting training samples in each iteration.

5.7. Resilience to feature perturbation

The reliability of GNNs in the presence of noisy graph data is a significant concern, especially considering the baseline datasets we employed. Graph data often contains inaccuracies or uncertainties in node attributes and edge connections. These inaccuracies can manifest as missing, incorrect, or redundant edges, as well as corrupted, incomplete, or irrelevant node attributes. Such issues challenge the performance and robustness of GNNs, which rely on the message-passing mechanism for information dissemination.

In this research, we evaluated the resilience of our GATE-GNN model compared to other methods, specifically focusing on handling feature noise (Abu-El-Haija et al., 2020; Juan et al., 2023; Li et al., 2024; Shi et al., 2021; Yang et al., 2023). We simulated feature noise by randomly eliminating node features, a situation that can naturally occur in Citation Network datasets, where crucial terms in article abstracts might be inadvertently omitted.

Our experiment involved comparing the performance of GATE-GNN, PMLP, INS-GNN, Boosting-GNN, GNN-CL, GCN, GraphSAGE, and GAT models under conditions of selective attribute elimination. The results are visually represented in Fig. 8. Notably, PMLP and INS-GNN performance show enormous resilience to feature perturbation. However, the proposed GATE technique's performance, which involved transferring weights between nodes in the weight domain and removing less relevant nodes or edges, was notably superior. As the proportion of eliminated features increased, the GATE technique's performance outshined that of the existing GNN models.

Fig. 8 demonstrates that our strategy, which integrates GATE with transfer learning, outperforms baseline methods in terms of accuracy and stability, even with increased feature noise. The GATE-GNN model's adaptability is highlighted by its ability to partially reconstruct deteriorated features, leveraging information from neighbouring nodes. This demonstrates not only the model's resilience to feature noise but also its robustness in maintaining high classification accuracy under challenging conditions.

5.8. Training time

In this study, we conducted a comparative analysis of training times among different models, focusing on our GATE-GCN model, PMLP, INS-GNN, GNN-CL, M-GCN, and Boosting-GCN. Additionally, we examined the impact of incorporating transfer learning in both GATE-GCN and Boosting-GCN models.

This comparison utilised the Cora dataset and examined models with varying numbers of classifiers (5, 7, and 9). The detailed training times for each setup are presented in Table 4. The GNN-CL model achieves the best performance without transfer learning. Our findings reveal that the GATE-GCN model, without transfer learning, achieves

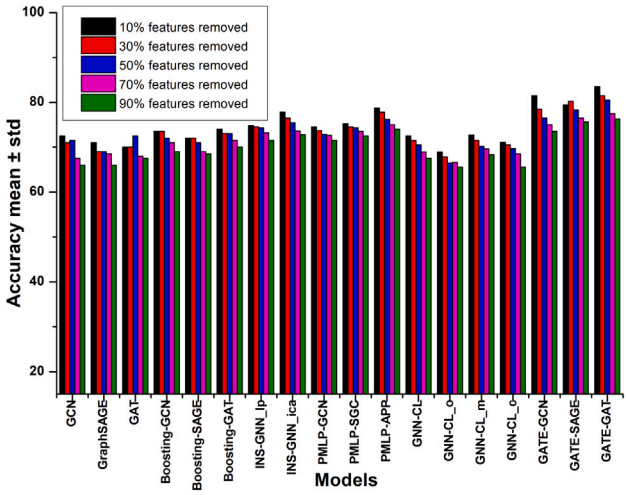


Fig. 8. Classification accuracy comparison under feature perturbation on the Cora dataset. This graph displays the results of experiments conducted on various GNN models. Each model was run 10 times, with the average accuracy calculated. Different random seeds were used in each run to ensure a variety of features were removed from the nodes, maintaining consistency in the random seeds across all tested GNN models. The graph highlights the comparative resilience of these models to feature noise.

Table 4
Training time (in seconds) for various methods with different classifier quantities. (t) denotes methods with transfer learning.

Model	Number of Classifiers		
	5	7	9
M-GCN	29.74 s	40.61 s	52.12 s
Boosting-GCN_t	11.34 s	14.43 s	20.11 s
Boosting-GCN	20.36 s	31.34 s	35.52 s
PMLP	10.56 s	15.61 s	20.45 s
GNN-CL	9.89 s	12.11 s	18.15 s
GNN-CL_O	13.56 s	15.31 s	21.95 s
GNN-CL_M	12.45 s	13.81 s	15.15 s
GNN-CL_C	15.16 s	19.22 s	26.65 s
INS-GNN_{ica}	115.25 s	121.61 s	123.67 s
INS-GNN_{lp}	106.66 s	110.91 s	118.85 s
GATE-GCN_t	7.45 s	9.46 s	12.54 s
GATE-GCN	12.57 s	16.78 s	21.52 s

NB: The top performance models are indicated in bold.

the second best, and with transfer learning performs the best. It consistently outperforms the other models in terms of training efficiency.

Compared to the existing five models, our GATE-GCN model demonstrates a notable reduction in training time. We utilised M-base classifier methods as benchmarks to compare the training efficiency of our GATE-GNN model using the GCN technique. All models were trained under identical settings on the Google Compute Engine Backend, equipped with a GPU and 12.68 GB of RAM. The superior training time of the GATE-GCN model underscores its effectiveness and efficiency in processing GNN tasks.

5.9. Effect of imbalance ratio

In this section, we assess the efficacy of GATE-GNN using various imbalance ratio (IR) techniques to ascertain the model's resilience to varying degrees of the imbalance issue. The experiment is performed on Cora dataset by systematically adjusting the IR from 0.2 to 0.6 while keeping all other variables unchanged. The IR represents the proportion of samples belonging to the minority class compared to the samples

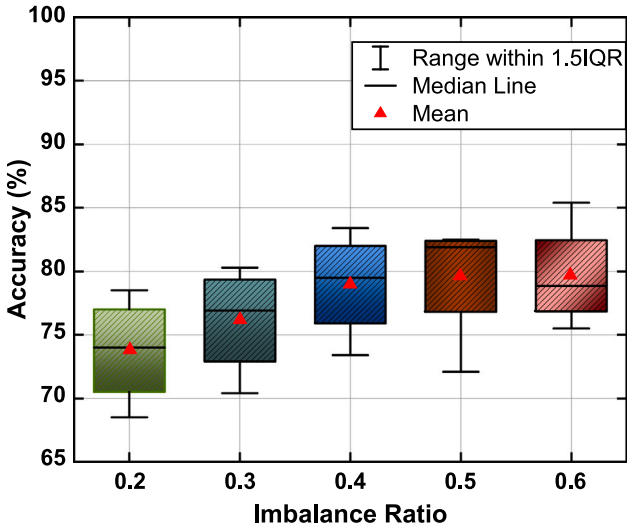


Fig. 9. Performance of the proposed GATE-GNN for imbalanced node classification at various imbalance ratios.

belonging to the majority class. A decrease in the IR results in a reduction in the number of samples belonging to the minority class. Fig. 9 displays the experimental findings. The whiskers of boxplots represent the range of accuracies observed over five experiments, while the line (red triangle) inside the boxplots indicates the mean value of classification accuracies. The performance of the proposed GATE-GNN grows as IR improves. This phenomenon may be attributed to the fact that incorporating a larger number of minority-class samples might enhance the effectiveness of the GATE-GNN technique in addressing imbalanced node classification tasks. This, in turn, leads to improved performance by mitigating bias and enhancing the classification accuracy of GNFE samples and related edges.

Five tests were performed on the Cora dataset, and the confusion matrix shows the average outcomes of the experiments for the proposed GATE-GNN model (Fig. 10). The classifier tends to allocate samples to the majority class due to the sample imbalance, resulting in the last column of the confusion matrix typically having the highest value (shown in the brightest colour). GATE-GNN enhances performance, particularly on the Cora dataset. The diagonal values of the confusion matrix increase, while the value in the last column drops due to the weight aggregation of base estimators. It combines several GNN classifiers to mitigate overfitting to some extent.

5.10. Analysis of parameter sensitivity

This section outlines the key hyperparameters in the GATE-GNN proposed model, which include λ_1 and λ_2 as defined in Eq. (22). These hyperparameters determine the relative importance of the objective function of h_t+h_v and highlight the significance of balancing competing goals in the classification aim. We do sensitivity analyses to assess the proposed approach using various values of λ_1 and λ_2, ranging from 0 to 0.1 and 0 to 0.01, respectively. This allows us to investigate the impact of the weighting hyperparameters. The experiment is performed in a controlled environment using the Cora dataset. The IR is set to 0.5, and the GATE-GAT classifier is used. Every experiment is conducted five times, and the average accuracies are presented in Table 5. When the value of λ_1 is set to 0, the accuracy improves as the value of λ_2 increases. The model's accuracy increases by around 10% when λ_2 changes from 0 to 0.01, compared to its accuracy at $\lambda_2 = 0$. A similar phenomenon arises when the value of λ_2 is held constant at 0 while λ_1 is varied. These findings indicate that each loss function has a role in achieving the optimisation target.

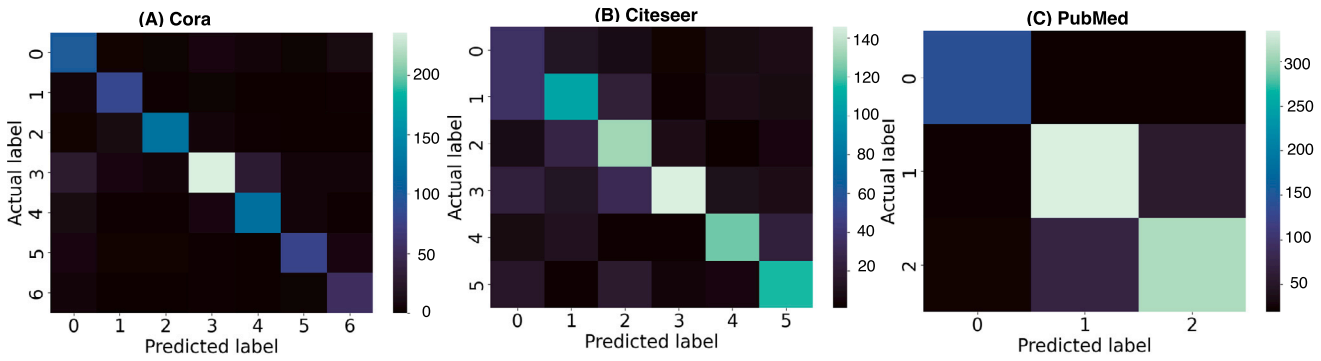


Fig. 10. Confusion matrix on Cora, Citeseer, and PubMed datasets with 36 labelled nodes for majority class and five labelled nodes for the minority class. (A) GATE-GCN on Cora, (B) GATE-GCN on Citeseer, and (C) GATE-GCN on PubMed.

Table 5
Sensitivity analysis of the parameters of loss components in terms of accuracy(in %).

λ_1	λ_2					
	0	0.002	0.004	0.006	0.008	0.01
0	75.4	79.3	76.5	77.4	81.9	82.3
0.02	78.5	79.5	78.4	79.5	80.5	82.4
0.04	79.6	80.5	81.4	81.3	82.4	81.9
0.06	79.4	80.6	81.3	81.2	81.3	**83.2**
0.08	79.5	80.4	81.4	81.5	**82.5**	82.1
0.1	79.3	80.3	81.2	82.1	**82.6**	**82.5**

NB: The top performance of λ are indicated in bold.

However, when both λ_1 and λ_2 are changed at the same time, the overall accuracy does not improve, even if both λ_1 and λ_2 are increased simultaneously. When the value of λ_2 is set to 0.01, increasing the value of λ_1 leads to an initial improvement in the model's performance, followed by a subsequent decline. This indicates that transfer learning approaches can optimise the alignments between two or more domains, and there is a trade-off for the allocated ensemble weights in λ_1 and λ_2 in tasks involving imbalanced node classification. To enhance flexibility, it is advisable to assign equal weight to λ_1 and λ_2 and fine-tune λ_1 for improved outcomes, given that λ_1 is set at 0.08 and λ_2 is set at 0.01.

In tests to see how sensitive the model is to parameter loss components, we found that when the number and type of attention heads is raised above 8, the model becomes less sensitive to the choice of the source and target domains and the number of attention heads. This makes the model overfit.

5.11. Analysis of model complexity

In this subsection, we experiment with model complexity based on floating-point operations (FLOPs), model size, and their influence on prediction accuracy and computational performance.

5.11.1. Model FLOPs and parameters

To thoroughly assess the balance between performance and complexity, we provide the floating-point operations (FLOPs) and model size for both the baselines and our model using the same hardware settings on Cora with the GCN classifier. In deep learning, FLOPs quantify the complexity and computational cost of a model, representing the number of floating-point operations required for tasks such as training or inference. The experiment was conducted using Python (torch framework) with GigaFLOPS (GFLOPS): 10^9 FLOPs. Table 6 displays the results. Calculating FLOPs for our model and the baseline involves analysing the operations within each layer of the network. For instance, in the GATE-GNN network, we compute the FLOPs by considering the number of multiplications and additions required for the GNN operations across the input data and filters in the GNN layer. Understanding and optimising FLOPs is essential for deploying models in real-time applications, especially on edge devices with limited computational resources, and for extending the battery life of battery-powered devices.

The results in Tables 2 and 6, as well as Fig. 11, indicate that GATE-GNN significantly enhances public datasets with minimal increase in model complexity. According to Table 6, GATE-GNN has the second-lowest number of parameters among the approaches that do not include transfer learning, whereas GNN-CL has the highest number of parameters. When utilising transfer learning, GATE-GNN achieves its best performance. Reducing the number of parameters notably lowers the expense of transferring the model for further research or implementation. However, GATE-GNN necessitates more floating-point operations and fewer floating-point computations when performed without transfer learning and with transfer learning, respectively. This outcome was achieved due to the advanced architecture, and: (1) The utilisation of feature extraction enables the model to adaptively learn from nodes and neighbouring base-learning based on labels, leading to an increase in computation; (2) The implementation of transfer learning and sparse connectivity reduces memory usage and effectively decreases the number of floating-point operations (FLOPs), resulting in an acceptable decrease in computational cost while maintaining model accuracy.

5.11.2. Model influence on prediction accuracy and computational performance

To further evaluate the learning process, we construct a decision tree classifier estimator using parameter values that incorporate the extracted features and target values. This approach enables us to gauge the complexity of the learning process. We calculate the average and standard deviation of the training and test sets, and determine accuracy ranges by applying a validation curve using a decision tree (Fig. 11). This curve plots the training and cross-validation scores (accuracy) for different depths of a decision tree classifier. The max-depth parameter controls the model's complexity, where a deeper tree (with a higher maximum depth) indicates a greater capability to capture complex patterns. GATE-GNN evaluates the importance of different neighbouring nodes near a given node and utilises this information in node feature propagation. This effectively reduces the influence of less important nodes within the graph during information transfer, but it increases model complexity without improving transfer learning. Consequently, the increased weight attention and transfer learning techniques enhance the model's efficacy and overall performance.

In our GATE-GNN model for node classification, we observe that as the maximum depth of the tree in Fig. 11 approaches 75, the model begins to optimise. This allows the tree to make up to 75 consecutive splits before making a prediction, capturing more subtle patterns in the data. Although in GNNs, unlike decision trees, the depth is not a direct parameter, the concept of model complexity remains highly

Table 6
Model FLOPs and parameters on Cora, (t) denotes methods with transfer learning.

Model	FLOPs(M)	Parameters(M)
M-GCN	3.4e+1	5.0e+0
Boosting-GCN_t	5.5e−1	6.2e−2
Boosting-GCN	5.5e+2	5.8e+0
PMLP	5.5e+2	5.9e+0
GNN-CL	5.6e−1	5.6e−2
GNN-CL_O	3.5e+1	5.3e+0
GNN-CL_M	3.2e+1	5.5e+0
GNN-CL_C	4.2e+1	6.5e+0
INS-GNN_{ica}	5.2e+2	9.6e+0
INS-GNN_{lp}	5.2e+1	9.2e+0
GATE-GCN_t	5.2e−1	5.8e−2
GATE-GCN	5.2e+2	6.2e+0

NB: The top performance models are indicated in boldface.

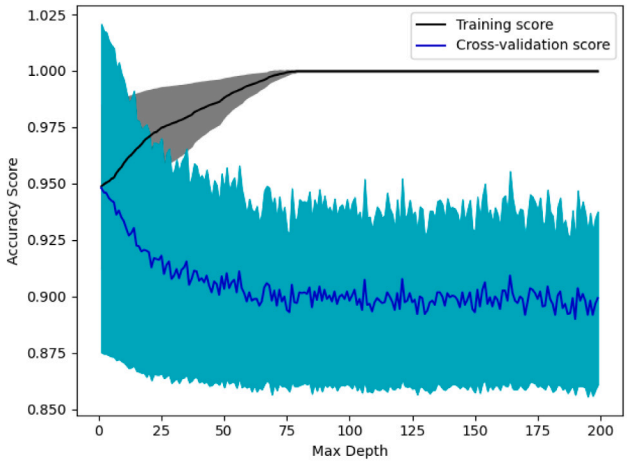


Fig. 11. Performance on different depths of a decision tree classifier.

relevant. Over-smoothing is a challenge in GNNs, where too many layers can make node features indistinguishable, analogous to an overly deep decision tree. To manage model complexity in our GATE-GNN effectively, it is crucial to strike a balance between the network's depth and the risk of over-smoothing. We aim for sufficient expressive power to accurately represent the complexities of the graph structure and node features. We employ transfer learning as a strategic approach to address the computational cost without compromising the model's performance.

6. Conclusion

This study introduced the GATE-GNN model as a solution to improve node classification in imbalanced graph neural networks (GNNs). The effectiveness of capturing spatial dependencies in graph data networks, which often struggle with high computational complexity and noisy edges that is affected by the traditional graph convolutional networks (GCNs). Similarly, despite the utility of enhancing node interactions in graph attention networks (GATs) and thus comes with increased computational demands.

By employing a graph node feature extraction technique that efficiently aggregates nodes, the GATE-GNN model addresses these challenges. This technique combines features of neighbouring nodes using adaptable weights, facilitating the construction of node embeddings well-suited for classification tasks. Additionally, by integrating GEWA connections that enable the model gains direct access to initial layer information. The GEWA leverages the spatial topology of GCNs to enhance feature extraction.

Our ensemble approach effectively minimises volatility and bias in individual datasets while optimising predictions and reducing errors. The implementation of transfer learning further aids in transferring features across graph network data and reducing computational costs. A notable novel in GATE-GNN is the addition of a GEWA self-attention mechanism in preprocessing the data. This filters out extraneous information and noisy edge features. GATE-GNN demonstrated state-of-the-art performance in imbalanced node classification when tested on benchmark datasets.

This research contributes valuable insights into the synergy between multilayer attention scores, graph node feature extraction (GNFE), node ensemble and transfer learning (NETL), and graph weight ensemble attention (GEWA). By harnessing the strengths of these techniques in spatial and temporal aspects of graph networks and algorithmic optimisation GATE-GNN shows promise in tackling classification challenges in imbalanced data scenarios. Although GATE-GNN achieved remarkable performance, it might introduce noise or bias from the auxiliary tasks, particularly if they are not well-aligned or consistent with the target task, which is dependent on hyperparameters and model initialisation. Looking ahead, future research will focus on the classification of generalisation in imbalanced higher-classifiers including specific minority classes (GIHC), improve the parameter sensitivity to choice of the source and target domains, and few-shot learning (FSL) methods within noisy graph networks. These forthcoming studies aim to develop novel algorithms and architectures to enhance the generalisation capabilities of models with limited labelled samples and addressing more complex classifier configurations in graph networks.

CRediT authorship contribution statement

Abdul Joseph Fofanah: Conceptualization, Formal analysis, Investigation, Methodology Writing, & editing, Writing – review. **David Chen:** Supervision, Funding acquisition, Investigation, Writing – review. **Lian Wen:** Supervision, Funding acquisition, Investigation, Writing – review. **Shaoyang Zhang:** Supervision, Funding acquisition, Investigation, Writing – review.

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.

References

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2020). N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *Uncertainty in artificial intelligence* (pp. 841–851). PMLR.

Acharya, D. B., & Zhang, H. (2020). Feature selection and extraction for graph neural networks. In *Proceedings of the 2020 ACM southeast conference* (pp. 252–255).

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., & Belongie, S. (2019). Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9268–9277).

Cui, Z., Zhang, J., Noh, G., & Park, H. J. (2022). MFDGCN: Multi-stage spatio-temporal fusion diffusion graph convolutional network for traffic prediction. *Applied Sciences*, *12*(5), 2688.

Douzas, G., & Bacao, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. *Information Sciences*, *501*, 118–135.

Fiorentini, N., & Losa, M. (2020). Handling imbalanced data in road crash severity prediction by machine learning algorithms. *Infrastructures*, *5*(7), 61.

Guo, D., Xu, C., & Tao, D. (2021). Bilinear graph networks for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems*.

Guo, Z., Zhang, C., Fan, Y., Tian, Y., Zhang, C., & Chawla, N. (2022). Boosting graph neural networks via adaptive knowledge distillation. ArXiv arXiv:2210.05920.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems: vol. 30*.

Han, X., Huang, Z., An, B., & Bai, J. (2021). Adaptive transfer learning on graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing* (pp. 878–887). Springer.

Huang, C., Li, Y., Loy, C. C., & Tang, X. (2016). Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5375–5384).

Ivanov, S., & Prokhorenkova, L. (2021). Boost then convolve: Gradient boosting meets graph neural networks. arXiv preprint arXiv:2101.08543.

Juan, X., Peng, M., & Wang, X. (2021). Exploring self-training for imbalanced node classification. In *Neural information processing: 28th international conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, proceedings, part v 28* (pp. 28–36). Springer.

Juan, X., Zhou, F., Wang, W., Jin, W., Tang, J., & Wang, X. (2023). INS-GNN: Improving graph imbalance learning with self-supervision. *Information Sciences*, *637*, Article 118935.

Junior, J. R. B., do Carmo Nicoletti, M., & Zhao, L. (2017). Attribute-based decision graphs: A framework for multiclass data classification. *Neural Networks*, *85*, 69–84.

Karagiannakos, S. (2021). Best graph neural network architectures: GCN, GAT, MPNN and more. *AI Summer*, *23*.

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys*, *52*(4), 1–36.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Li, X., Fan, Z., Huang, F., Hu, X., Deng, Y., Wang, L., et al. (2024). Graph neural network with curriculum learning for imbalanced node classification. *Neurocomputing*, *574*, Article 127229.

Li, B., & Nabavi, S. (2023). A multimodal graph neural network framework for cancer molecular subtype classification. arXiv preprint arXiv:2302.12838.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

Liu, M., Jin, S., Jin, L., Wang, S., Fang, Y., & Shi, Y. (2022). Imbalanced nodes classification for graph neural networks based on valuable sample mining. In *Proceedings of the 2022 6th international conference on electronic information technology and computer engineering* (pp. 1957–1962).

Maekawa, S., Noda, K., Sasaki, Y., et al. (2022). Beyond real-world benchmark datasets: An empirical study of node classification with GNNs. *Advances in Neural Information Processing Systems*, *35*, 5562–5574.

Maurya, S. K., Liu, X., & Murata, T. (2022). Simplifying approach to node classification in graph neural networks. *Journal of Computer Science*, *62*, Article 101695.

Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 971–987.

Pfeifer, B., Chereda, H., Martin, R., Saranti, A., Clemens, S., Hauschild, A.-C., et al. (2023). Ensemble-GNN: Federated ensemble learning with graph neural networks for disease module discovery and classification. (pp. 2003–2023). bioRxiv.

Puny, O., Ben-Hamu, H., & Lipman, Y. (2020). Global attention improves graph networks generalization. arXiv preprint arXiv:2006.07846.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93–93.

Shi, S., Qiao, K., Yang, S., Wang, L., Chen, J., & Yan, B. (2021). Boosting-GNN: Boosting algorithm for graph networks on imbalanced node classification. *Frontiers in Neurorobotics*, *15*, Article 775688.

Sun, C., Li, C., Lin, X., Zheng, T., Meng, F., Rui, X., et al. (2023). Attention-based graph neural networks: A survey. *Artificial Intelligence Review*, *56*, 2263–2310.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al. (2017). Graph attention networks. *Stat*, *1050*(20), 10–48550.

Wang, Y. (2022). Fair graph representation learning with imbalanced and biased data. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 1557–1558).

Wang, C., Chen, B., Li, G., & Wang, H. (2021). FL-AGCNS: federated learning framework for automatic graph convolutional network search. arXiv preprint arXiv:2104.04141.

Wang, R., Xiong, W., Hou, Q., & Wu, O. (2022). Tackling the imbalance for gnns. In *2022 international joint conference on neural networks* (pp. 1–8). IEEE.

Wang, Y., Zhao, Y., Shah, N., & Derr, T. (2022). Imbalanced graph classification via graph-of-graph neural networks. In *Proceedings of the 31st ACM international conference on information & knowledge management* (pp. 2067–2076).

Wei, Q., Wang, J., Hu, J., Li, X., & Yi, T. (2022). OGT: Optimize graph then training GNNs for node classification. *Neural Computing and Applications*, *34*(24), 22209–22222.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861–6871). PMLR.

Yang, C., Wu, Q., Wang, J., & Yan, J. (2023). Graph neural networks are inherently good generalizers: Insights by bridging GNNs and MLPs. In *International conference on learning representations*.

Zhang, M., Hu, Q., Sun, P., Wen, Y., & Zhang, T. (2023). Boosting distributed full-graph GNN training with asynchronous one-bit communication. arXiv preprint arXiv:2303.01277.

Zhao, M., Qiu, W., Wen, T., Liao, T., & Huang, J. (2021). Feature extraction based on gabor filter and support vector machine classifier in defect analysis of thermoelectric cooler component. *Computers & Electrical Engineering*, *92*, Article 107188.

Zhao, C., Wu, D., Huang, J., Yuan, Y., Zhang, H.-T., Peng, R., et al. (2022). BoostTree and BoostForest for ensemble learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhao, T., Zhang, X., & Wang, S. (2021). Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 833–841).

Zhou, Q., Chen, N., & Lin, S. (2022). FASTNN: A deep learning approach for traffic flow prediction considering spatiotemporal features. *Sensors*, *22*(18), 6921.