Neurocomputing 602 (2024) 128273

Contents lists available at ScienceDirect

Neurocomputing

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

WSSGCN: Wide Sub-stage Graph Convolutional Networks

Chao Wang [a,1,*], Zheng Tang [b,1], Hailu Xu [c]

[a] Department of Communication signal research institute, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China
[b] NVIDIA, Redmond, WA 98052, USA
[c] Department of Computer Engineering and Computer Science, California State University, Long Beach, CA, 90840, USA

ARTICLE INFO

Communicated by A. Iosifidis

Dataset link: https://doi.org/10.48550/arXiv.1611.08402, https://doi.org/10.1145/276675.276685, https://doi.org/10.1609/aimag.v29i3.2157, https://doi.org/10.48550/arXiv.2002.05287, https://doi.org/10.48550/arXiv.2206.08583

Keywords:
Graph convolutional networks
Convolutional neural networks
Sub-stage framework
Isomorphic graphs
Heterogeneous graphs
Wider GCNs

ABSTRACT

Graph Convolutional Networks (GCNs) have emerged as a potent tool for learning graph representations, finding applications in a plethora of real-world scenarios. Nevertheless, a significant portion of deep learning research has predominantly concentrated on enhancing model performance via the construction of deeper GCNs. Regrettably, the efficacy of training deep GCNs is marred by two fundamental weaknesses: the inadequacy of conventional methodologies in handling heterogeneous networks, and the exponential surge in model complexity as network depth increases. This, in turn, imposes constraints on their practical utility. To surmount these inherent limitations, we propose an innovative approach named the Wide Sub-stage Graph Convolutional Network (WSSGCN). Our method is an outcome of meticulous observations drawn from classical and graph convolutional networks, aimed at rectifying the constraints associated with traditional GCNs. Our strategy involves the conception of a staged convolutional network framework that mirrors the fundamental tenets of the step-by-step learning process akin to human cognition. This framework prioritizes three distinct forms of consistency: response-based, feature-based, and relationship-based. Our approach involves three tailored convolutional networks capturing node/edge, subgraph, and global features. Additionally, we introduce a novel method to expand graph width for efficient GCN training. Empirical validation on benchmarks highlights WSSGCN's superior accuracy and faster training versus conventional GCNs. WSSGCN triumphs over traditional GCN constraints, significantly enhancing graph representation learning.

1. Introduction

Graph-structured data finds wide-ranging applications across various domains, encompassing fields like social network analysis [1,2], computer vision [3,4], bioinformatics [5–7], and action recognition [8,9]. Among the arsenal of tools to glean insights and knowledge from relational data, Graph Convolutional Networks (GCNs) stand out as a potent force. Their prowess in extracting meaningful information from interconnected data has led to remarkable breakthroughs in a multitude of domains, including traffic forecasting [10,11], image classification [12,13], and visual reasoning [14]. A plethora of GCN variants [15–20] have been proposed, all with the common aim of molding low-dimensional representations for graph-structured data. Yet, in the pursuit of heightened accuracy, the trend has been to concoct graph model architectures with progressively deeper layers [21–23], inadvertently giving rise to challenges such as escalated training costs.

A plethora of methods based on Graph Convolutional Networks (GCNs) has recently emerged, encompassing both homogeneity [20,24–26] and heterogeneity [27–32]. However, real-world networks often exhibit inherent heterogeneity, posing challenges for many methods. Approaches may generate similar node representations assuming homogeneity or crudely aggregate distant neighbors to address heterogeneity. Unfortunately, this can lead to mixed node representations struggling to differentiate diverse classes. Moreover, deep graph neural networks' increased depth escalates memory consumption, potentially bottlenecking GPU memory and posing significant challenges. Notably, the sub-stage feature extraction method proposed in [33] is specifically tailored for conventional knowledge distillation models, with its primary focus on facilitating knowledge transfer between teacher and student models. However, its applicability remains confined and does not readily extend to the broader domain of graph neural networks.

To address these challenges, we propose a novel width-graph convolutional neural network (WSSGCN), designed to handle graph network structures effectively. Our approach is inspired by human step-learning [34,35], and we aim to build a width-based network model structure that is different from the prediction problems currently defined on individual nodes, pair-wise relationships, or entire graphs.

* Corresponding author.
E-mail addresses: chaowangasaph@gmail.com (C. Wang), tangzhengthomas@gmail.com (Z. Tang), Hailu.Xu@csulb.edu (H. Xu).
[1] Member, IEEE

https://doi.org/10.1016/j.neucom.2024.128273
Received 13 April 2024; Received in revised form 7 July 2024; Accepted 24 July 2024
Available online 29 July 2024

Specifically, we divide the WSSGCN into three sub-stages as shown in 1(b) tailored to the unique characteristics of graph networks, namely response-substage, feature-substage, and relationship-substage, and design three convolutional neural networks to embed the corresponding knowledge of each sub-stage. Finally, we integrate a combination module with different characteristics into a unified loss function based on the width principle, considering the different data conversions from the three networks.

Our contributions are three-fold:

- We introduce the Wide Sub-stage Graph Convolutional Network (WSSGCN), which adeptly learns informative structural features through the strategic expansion of network widths, simultaneously enhancing training speed.
- WSSGCN employs a heuristic stage-based learning approach, culminating in the creation of three distinct sub-stage learning structures, each accompanied by its dedicated convolutional network channels.
- Our extensive experimentation on benchmark graph datasets showcases the prowess of the WSSGCN architecture in mitigating computational and memory challenges associated with large graphs, yielding either state-of-the-art or highly competitive results.

2. Related works

Isomorphic Graphs. Convolutional Neural Networks (CNNs) were first introduced by Scarselli et al. [36] as an extension of neural network models for processing graph-based data. This pioneering work ignited a keen interest in deep graph learning within the machine learning and computer vision communities. Nevertheless, Graph Neural Networks (GNNs) face model expression limitations, hindering their grasp of graph topology. Subsequent work [37–40] aims to enhance GNNs by improving structural information. Kipf and Welling [24] introduced Graph Convolutional Networks (GCNs), scalable for semi-supervised learning with graph-structured data. Further efforts by Rahimi et al. [41], Defferrard et al. [25], and Colombo et al. [42] propose techniques to extend GCNs. However, generalizing these methods for heterogeneous graphs remains a challenge, underscoring the vast potential of GNNs and the need for ongoing research.

Heterogeneous Graphs. Heterogeneous information networks are intricate structures comprising diverse interacting components, furnishing an abundance of structural and semantic insights beyond those offered by isomorphic networks. Over recent years, researchers have devised an array of structural analysis techniques for dissecting these networks. Among them, message-passing neural networks (MPNNs) stand out. Introduced by Gilmer et al. [43], MPNNs adeptly encode atomic and bond properties such as type and order, facilitating message-passing and node representation updates. This empowers MPNNs to capture intricate interactions between atoms and bonds in molecules, irrespective of their classifications or connections. CPGNN by Zhu et al. [27] enriches GNNs with a compatibility matrix for diverse node connections. Wang et al. [29] use attention mechanisms for neighboring nodes, considering various features. Yan et al. [30] propose the "heterophily-adjusted degree" for enhanced GCN architectures. However, these models face challenges in generalizing across heterogeneous network applications.

Deeper GCNs. Traditional GCNs face over-smoothing and representation issues, hindering distant node distinction and structural insight. Numerous strategies have emerged to surmount these limitations. Chiang et al. [44] introduced Cluster-GCN, which segments the graph into smaller subgraphs via an innovative clustering algorithm. "MixHop" [45] blends multi-scale insights, and Li et al. [23] use residual connections and normalization for deep GCNs. Li et al. [21] introduced "RevGNN" for gradient-based training of deep GNNs, but these techniques trade complexity for improved performance. Their effectiveness is often dataset-specific, challenging generalization to diverse scenarios.

3. Proposed method

We first supply the notations & definitions of the GCN method to inspire the proposed algorithm.

3.1. Preliminaries

Notation. Heterogeneous graphs [46] combine various object classes with different relationships between them. Let G (V, E) as a heterogeneous graph, where the node type mapping function is identified as $\Phi : V \rightarrow N$ and the edge type mapping function is designated as $\Psi : E \rightarrow P$. The sets N and P represent predetermined object types and link types where the sum of their cardinalities surpasses two. The heterogeneous graphs adjacency matrix is denoted as A, with the degree of node $vi \in V$ being signified as $\hat{d}i$, and the degree matrix, represented by a diagonal matrix, consists of the degrees of each node denoted by $\hat{D}$.

GCNs. GCN [24] has become a famous graph neural network architecture due to its simplicity and effectiveness in various tasks and applications. Specifically, we consider a multilayer graph convolutional network (GCN) with the following hierarchical propagation rules:

$$H^{k+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{k}W^{k}), \tag{1}$$

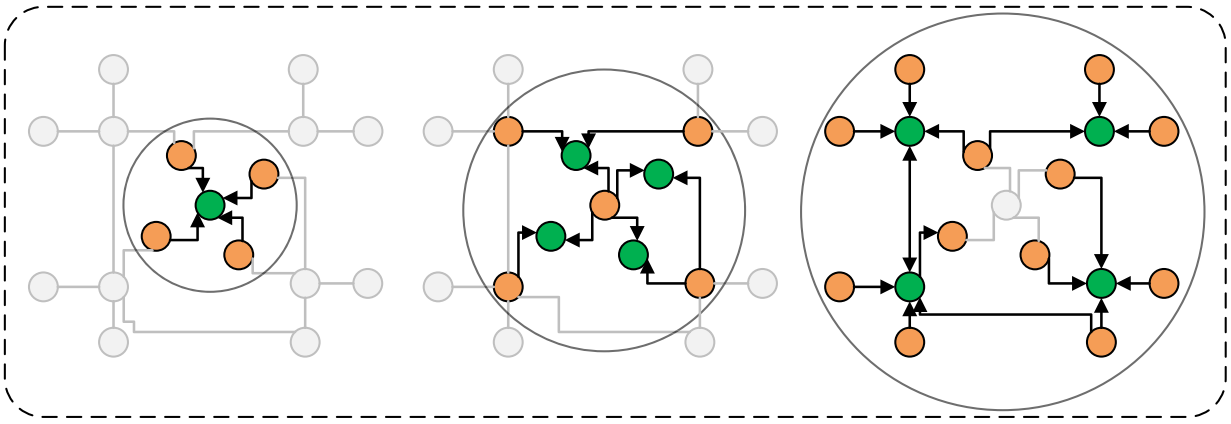
where $\hat{A} = A + I$ is the adjacency matrix of undirected graph G plus self-joining, which allows the node representation itself to be included when updating the node representation. $I \in \mathbb{R}^{N\times N}$ is the identity matrix. k is the layer of the network. $\sigma(\cdot)$ is the activation function, such as ReLU and Tanh.

3.2. Substage-based heuristic methods

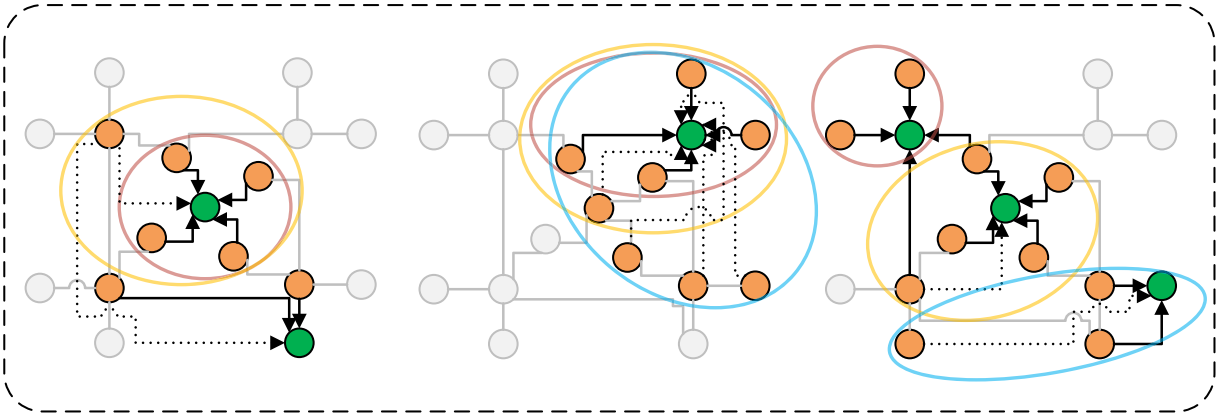
Drawing inspiration from [34,35], we have introduced a graph neural network structure that reflects the sub-stages of human learning. To begin with, [34] introduces Piaget's stages of cognitive development in children, covering infancy to adolescence. Each stage presents distinct cognitive characteristics and developmental tasks. For instance, in the Response-based substage (similar to our Response-based substage), infants learn through sensory exploration, while in the Feature-based substage (similar to our Feature-based substage), basic symbolization is involved. The subsequent Concrete Operational and Formal Operational stages (similar to our Relation-based substage) introduce more intricate abilities like logical reasoning. Piaget's theories provide valuable guidance for curriculum design, assessment, and enhancement. For instance, educators can tailor teaching methods and strategies based on students' age groups. For younger students, specific imaginative methods such as games and crafts can be employed (Response-based substage). For older students, more abstract and investigative methods, like problem-solving activities, are more suitable (Feature-based substage and Relation-based substage).

On the other hand, [35] emphasizes that learning is an active process, where learners must engage with real-life contexts and meaningful experiences to deepen their comprehension and application of knowledge. This aligns with Piaget's theories, as both underscore the significance of learners' active participation and experiences throughout the phased learning process.

Based on these theories and research findings, we formulated the concept of an analogous "human learning" process. We then proceeded to model the structural information of the human learning process. To achieve this objective, we developed three sub-stage designs that are not only intuitive but also remarkably efficient. For a visual representation of these designs, please refer to 1(b).



(a) Traditional depth graph convolution process.



(b) Substage process.

Fig. 1. Illustration of three substages: Green nodes calculate current features, orange nodes are connected to green nodes, and gray nodes are unrelated. Solid black lines denote edges to green nodes, while dotted lines represent potential relations. WSSGCN differs by extracting width-based sub-stage features, not relying on depth-based central node expansion. Red circle: response-based information transfer. Yellow circle: feature-based information transfer. Blue circle: relationship-based information transfer indicating graph-wide relationships.

3.2.0.1. Response-based substage. Similar to the gradual learning process of humans, the response-based substage is designed to calculate the contributions of peripheral nodes that are directly related to the nodes. Our approach involves a one-hop-centric coding method that assigns an embedding vector to each central node based on its indegree and outdegree. This central code is then applied to every node and added as an input to the node characteristics, allowing for a more intuitive representation of the data. The Eq. (1) contains multiple nodes, and each node can be expanded upon as:

$$H_i^{k+1} = \sigma\Big(\sum_{j\in N(i)} \frac{A_{ij}}{D_{ii}^{\frac{1}{2}} D_{jj}^{\frac{1}{2}}} H_j^k W^{k+1} + \frac{1}{D_i} H_i^k W^{k+1}\Big), \tag{2}$$

As shown in Eq. (2), the characteristics of each node in layer K + 1 are mainly decomposed into two parts. In order to generalize and learn these structural features, we propose feature generator functions F_{node}, F_{edge}, and $F_{response}$ to learn to generate the optimal features only through the adjacency matrix $\hat{A}$ as:

$$\theta_i^{response} = F_{node}\Big(\sum_{j\in N(i)} F_{edge}(A_{ij}) + F_{response}\Big), \tag{3}$$

where $\theta_i^{response}$ is the learned representation of the response-based node, and F_{node}, F_{edge}, and $F_{response}$ are three eigenfunctions composed of learnable functions. The adjacency matrix A is input to generate the most favorable structural features. After establishing the connection of a node (i, j), we calculate the prediction score from each representation matrix. To ensure accuracy, we adjust the learning weights based on the response-based substage through the trainable parameter α, as follows:

$$\hat{y}_{ij}^{response} = \alpha \cdot \sigma((\theta_i^{response})^T \theta_j^{response}) + (1-\alpha) \cdot (KL((\varphi_i^{response})^T \parallel \varphi_j^{response})), \tag{4}$$

where $KL(\cdot)$ is the KL divergence used to measure the consistency of the distribution between $(\theta_i^{response})^T$ and $\theta_j^{response}$, and α is a hyperparameter that balances the impact of KL divergence. $\varphi_i^{response}$ and $\varphi_j^{response}$ are respectively the softmax transformations for each feature dimension of $\theta_i^{response}$ and $\theta_j^{response}$, ensuring that all values satisfy the requirements of a probability distribution in the formalism.

3.2.0.2. Feature-based substage. The feature-based subphase is akin to the way humans learn on a local scale. Its purpose is to determine the significance of nodes within a multi-hop range surrounding the node. Our approach utilizes a multi-hop center coding method that allocates embedding vectors to various nodes based on their in-out and in-out degrees. These vectors are then added as inputs to the central node features, resulting in a comprehensive representation of the data locally. Eq. (5) comprises multiple nodes, each of which can be further

elaborated as follows:

$$H_i^{k+1} = \sigma(\psi(\sum_{j\in N(i)} \eta^k \frac{A_{ij}}{D_{ii}^{\frac{1}{2}} D_{jj}^{\frac{1}{2}}} H_j^k W^{k+1} + \frac{1}{D_i}\eta^k H_i^k W^{k+1})), \quad (5)$$

where η is a hyperparameter that regulates the weights of both the nearer and farther neighbors, while ψ is a multi-layer perceptron that manages the scale of H_i^{k+1}, to learn and generalize these multi-hop local range structural features effectively, we introduce the feature generator function $_{F_{feature}}$, which learns to generate optimal features solely through the adjacency matrix $\hat{A}$ as:

$$\theta_i^{feature} = F_{node}(\psi(\sum_{j\in N(i)} \eta_{F_{edge}}(A_{ij}) + \eta_{F_{feature}})), \quad (6)$$

where $\theta_i^{feature}$ is the learned representation of the feature-based node, and $_{F_{feature}}$ is the eigenfunction composed of learnable functions. Through a tunable parameter β, we amend the learning weights in proportion to the feedback-defined substage to ensure exactness as follows:

$$\hat{y}_{ij}^{feature} = \beta \cdot \sigma((\theta_i^{feature})^T \theta_j^{feature}) + (1-\beta)\cdot(KL((\varphi_i^{feature})^T \parallel \varphi_j^{feature})), \quad (7)$$

3.2.0.3. Relation-based substage. Relationship-based substages are analogous to human perception of the overall scope of things. Its purpose is to determine the importance of having an express representation around that node globally. Our method uses multi-hop center relation coding and assigns the embedded vectors to each node according to the importance of global and local relations. These vectors are then added to the central node features as inputs to represent the global data comprehensively. Eq. (8) contains many nodes, each of which can be further elaborated as follows:

$$H_i^{k+1} = \sigma(\psi(\sum_{j\in N(i)} \mu^k \frac{A_{ij}}{D_{ii}^{\frac{1}{2}} D_{jj}^{\frac{1}{2}}} H_j^k W^{k+1}) + \phi(\sum_{j\in N(i)} \frac{1}{D_j} \nu^k H_j^k \parallel W^{k+1})), \quad (8)$$

where μ and ν are two hyperparameters that mediate both global and local relation representation weights, and ϕ is a multi-layer perceptron that manages the scale of global relation, to effectively learn and generalize the global and local relation representations, we introduce the relation eigenfunction $_{F_{relation}}$, which learns relation only through the adjacency matrix A, as follows:

$$\theta_i^{relation} = F_{node}(\psi(\sum_{j\in N(i)} \mu_{F_{edge}}(A_{ij}) + \phi(\sum_{j\in N(i)} \nu_{F_{relation}}))), \quad (9)$$

where $\theta_i^{relation}$ is the learned representation of the relation-based node. With one tunable parameter γ, we correct the learning weights in proportion to the sub-stages defined by feedback to ensure accuracy, as follows:

$$\hat{y}_{ij}^{relation} = \gamma \cdot \sigma((\theta_i^{relation})^T \theta_j^{relation}) + (1-\gamma)\cdot(KL((\varphi_i^{relation})^T \parallel \varphi_j^{relation})). \quad (10)$$

3.2.0.4. Total loss function. Our framework employs a set of three binary cross-entropy losses to optimize the performance of the three sub-stage models we have proposed, according to information from (4), (7), and (10).

$$L = \sum_{(i,j)\in\hat{D}} \rho_1 \Gamma(\hat{y}_{ij}^{response}, y_{ij}) + \rho_2 \Gamma(\hat{y}_{ij}^{feature}, y_{ij}) + \rho_3 \Gamma(\hat{y}_{ij}^{relation}, y_{ij}), \quad (11)$$

where, the weights, denoted by ρ_i, and binary cross-entropy loss, represented as $\Gamma(\cdot,\cdot)$.

3.3. Width-based graph convolutional networks

We introduce the Width-based Graph Convolutional Networks (WSSGCN) method, as depicted in Fig. 2, which emphasizes graph convolutional network width, akin to [47], for proficiently capturing structural insights from input graphs. In contrast to conventional graph convolutional networks, WSSGCN incorporates neighborhood width, leveraging a trainable width-based kernel to weigh node contributions according to their proximity to the central node. This adaptive approach allows WSSGCN to tailor its features to the input graph's characteristics through kernel learning during training.

WSSGCN extracts multiple sets of representation matrices from three substages spanning the width dimension. These matrices are grouped to capture corresponding feature dimensions and subsequently combined to construct the ultimate vector for the fully connected layer. Augmenting the architecture, a pooling layer amalgamates node features into a singular feature vector. By selecting the node with the highest weight based on the width-based kernel, this pooling mechanism accommodates graphs of varying sizes and substantially enhances WSSGCN's performance.

3.4. Discussion of the proposed method

3.4.0.1. Basic combination. WSSGCN employs the substage heuristic method, which comprises three straightforward and efficacious substage designs. The model offers notable benefits in effectively capturing the structural characteristics of human learning processes, accommodating diverse node and relationship types, and integrating extensive semantic information within heterogeneous graphs. Additionally, the model leverages the GCN architecture and width propagation rules to compute the contributions of peripheral nodes directly connected to the primary nodes. The comprehensive computational procedure of the WSSGCN algorithm is illustrated in Algorithm 1.

Algorithm 1 WSSGCN pipeline.

Input:
adjacency matrix $\boldsymbol{A}$, Convergence step $\boldsymbol{S}$,
Number of layers $\boldsymbol{K}$
Output:
Graph feature matrix θ^{final}
Stage 1: Initial WSSGCN
Initialize hyperparameter: α, β, γ, η, μ, ν, ρ_1, ρ_2, ρ_3
Stage 2: Feature Convergence
for $s = 1; i \le S$; **do**
  for $n = 1; i \le N$; **do**
    Revise the standardized adjacency matrix $\hat{\boldsymbol{A}}$
    for $k = 1; i \le K$; **do**
      Calculate $\theta_n^{response}$, $\theta_n^{feature}$, *and* $\theta_n^{relation}$ with Eq. 3, Eq. 6, and Eq. 9
      Compute the θ_n^{final} with $\theta_n^{response}$, $\theta_n^{feature}$, and $\theta_n^{relation}$
    end for
  end for
end for
Stage 3: Feature Ensemble
Calculate feature matrix $\theta^{final} = \frac{1}{n}\sum_{i=1}^{n} \theta_i^{final}$

3.4.0.2. Complexity. Our algorithm exhibits high efficiency and parallelizability. When considering the standard WSSGCN feature representation transfer, the time complexity can be expressed as O(N,P), which is determined by the maximum value among the following three terms:

$$O(N,P) = max(\sum_{j\in N(i)} O(\Phi_{response}(A_{ij}) + \Psi_{response}(A_{ij})),$$

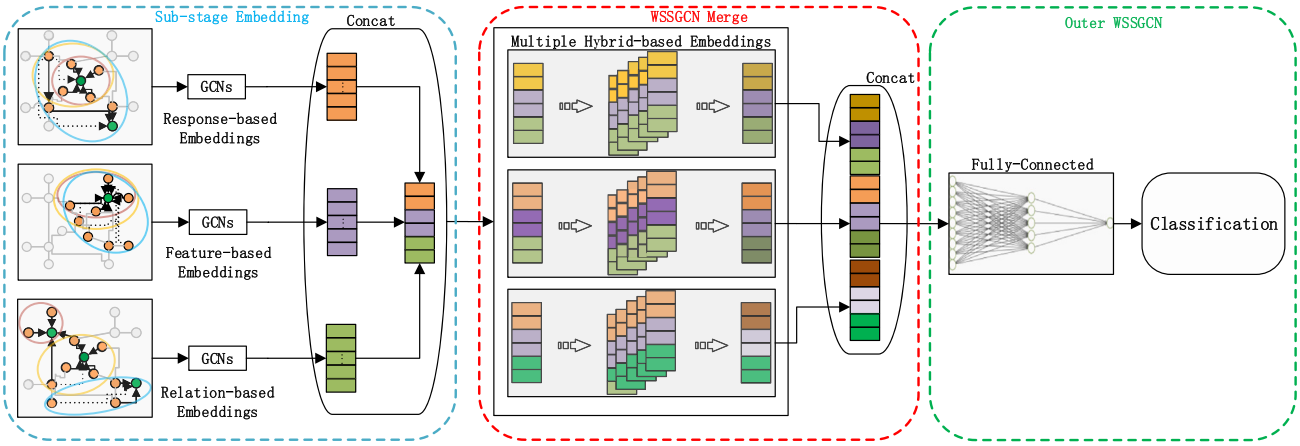


Fig. 2. WSSGCN process. WSSGCN has three stages: feature extraction, hybrid embedding fusion, and classification prediction. In the feature extraction stage, width vectors are synthesized via three sub-stages. The hybrid embedding fusion stage combines width vectors into a unique representation matrix. Classification prediction is done through a fully connected layer.

$$\sum_{j\in N(i)} O(\Phi_{feature}(A_{ij}) + \Psi_{feature}(A_{ij})),$$
$$\sum_{j\in N(i)} O(\Phi_{relation}(A_{ij}) + \Psi_{relation}(A_{ij}))), \quad (12)$$

where N represents the set of node types, and P represents the set of edge types. Φ and Ψ represent sub-stage matrix feature extraction and matrix KL divergence calculation functions respectively. The overall complexity of the WSSGCN model is linear concerning the number of node and edge types. The model comprises three subphases, allowing easy parallelization of the response-based, feature-based, and relation-based components. More convergence analysis is provided in 4.5.

4. Experiments

We have designed experiments to evaluate the efficacy of the WSSGCN classification framework that we have proposed. Our analysis involves evaluating the contributions of each component of our model toward generating homogeneity and heterogeneity graphs. We also aim to compare the performance of our model with state-of-the-art models on the benchmark to verify our approach's superiority.

4.1. Experimental setup

4.1.0.1. Datasets. It is common to distinguish between heterogeneous and homogeneous graphs to analyze real-world graphs. We have used six heterogeneous graphs, namely Texas, Wisconsin, Squirrel, Actor, Cornell, Chameleon, Wiki and ogbn-arxiv [28,48–53], and three homogeneous graphs, Cora, Pubmed, and Citeseer [54–56]. Table 1 provides summary statistics for these graphs.

4.1.0.2. Baselines. To evaluate the effectiveness of WSSGCN, we conduct a comparative analysis against five classical heuristic GNN methods, namely vanilla GCN [24], GAT [26], GraphSage [17], NAFS [57] and GDGNN [58]. Additionally, we compare it with two smoothing models, PairNorm [59] and GCNII [60], and six methods for dealing with heterogeneous models, including CPGNN [27], GGCN [30], MixHop [61], Geom-GCN [28], GPRGNN [62], and H_2GCN [63]. For baseline models with multiple variants (CPGNN [27], Geom-GCN [28], GCNII [60], NAFS [57], H_2GCN [63]), we select the best variant for each dataset. All experiments are conducted using the PyTorch framework on an NVIDIA Tesla P100 industrial GPU server. The original code provided by the author is used for all experiments.

For hyperparameter settings, we conduct a hyperparameter tuning process for the WSSGCN model in our experiments. We test various values for the following hyperparameters:

Table 1
The stats for our actual graphs include node count, edge count, classes, and homophily proportion h.

Dataset	Nodes	Edges	Classes	h
Cora	2708	5278	6	0.8
Citeseer	3327	4676	7	0.74
Pubmed	19,717	44,327	3	0.85
Cornell	183	280	5	0.3
Wisconsin	251	466	5	0.21
Actor	7600	26,752	5	0.22
Texas	183	295	5	0.11
Chameleon	2277	31,421	5	0.23
Squirrel	5201	198,493	5	0.22
Wiki	2405	17,981	17	0.9
ogbn-arxiv	2,449,029	61,859,140	47	2.6

- Dropout rate: a range between 0.0 and 0.5
- Weight decay: a range between 1e–6 and 1e–2
- Hidden units: the possible values are 8, 16, 32, 64, and 128
- Decay rate: a range between 0.0 and 1.0

We use the hyperparameters specified by their authors in their papers for the other models. All models are optimized using the Adam optimizer. The GAT model uses an initial learning rate of 0.005, while the Geom-GCN model employs a custom learning scheduler. All other models, including WSSGCN, use an initial learning rate of 0.01. Our meticulous approach to hyperparameter tuning allows us to optimize the performance of the WSSGCN model and ensure that all other models are operating at their peak potential.

For WSSGCN, the initialization parameters which are slightly different according to the complexity of the data sets, as shown in Table 2.

4.2. Comparison of training accuracy

4.2.0.1. Performance on homogeneity and heterogeneity. Table 3 presents the outcomes of our experiments, where we assessed diverse graph neural network (GNN) models using datasets of varying levels of homophily and heterogeneity. We present the test accuracy of each model across different layers, focusing on the task of supervised node classification. Specifically, we highlight the best performance achieved by each model across all layers.

It is important to highlight that WSSGCN does not consistently attain optimal performance in either homogeneous or heterogeneous datasets. This behavior can be attributed to the dataset's limited capacity, which may not fully demonstrate the advantages of WSSGCN's stage learning across diverse homogeneous or heterogeneous scenarios.

Table 2
Initialize parameter weights for different datasets.

	Citeseer	Pubmed	Cora	Texas	Squirrel	Chameleon	Wisconsin	Cornell	Actor	Wiki	ogbn-arxiv
α	0.5	0.5	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.5
β	1.0	1.0	1.0	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.5
γ	1.5	1.5	1.5	2.0	2.0	2.0	2.0	2.0	2.0	2.0	6.0
ρ_1	1.0	1.0	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	4.0
ρ_2	2.5	2.5	2.5	4.0	4.0	4.0	4.0	4.0	4.0	4.0	6.0
ρ_3	1.0	1.0	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	4.0
η	2.0	2.0	2.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.5
μ	1.0	1.0	1.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	12.0
ν	0.5	0.5	0.5	10.0	10.0	10.0	10.0	10.0	10.0	10.0	30.0

Table 3
Accuracy ± stdev on graphs with heterophilic and homophilous features. We report top performance across layers. Highlighted models are consistently best for each benchmark. Gray background highlights second-ranked models. Benchmarks are based on 10 splits (48%:32%:20% nodes per category for train/validation/test) from [28].

Models	Homogeneity			Heterogeneity							
	Citeseer	Pubmed	Cora	Texas	Squirrel	Chameleon	Wisconsin	Cornell	Actor	Wiki	ogbn-arxiv
WSSGCN	**80.25 ± 1.35**	**92.76 ± 0.36**	88.39 ± 0.48	83.90 ± 4.67	**56.67 ± 1.36**	**72.86 ± 2.60**	**86.40 ± 4.95**	83.57 ± 5.96	**40.57 ± 1.10**	**57.87 ± 0.68**	**77.26 ± 1.39**
GCN	75.32 ± 0.67	88.27 ± 0.48	83.62 ± 0.36	56.91 ± 4.75	55.10 ± 1.10	65.83 ± 1.90	52.39 ± 3.18	62.07 ± 4.25	28.12 ± 1.07	48.70 ± 2.29	70.93 ± 3.26
GAT	75.51 ± 1.25	84.97 ± 1.36	87.69 ± 1.15	53.42 ± 5.16	41.92 ± 1.24	61.39 ± 1.48	50.61 ± 3.55	62.77 ± 4.93	26.55 ± 1.39	49.23 ± 1.72	71.26 ± 2.18
GraphSage	75.07 ± 0.58	88.12 ± 1.71	87.35 ± 1.68	83.76 ± 5.51	40.06 ± 0.34	56.16 ± 0.97	82.70 ± 4.48	77.02 ± 4.39	35.26 ± 0.41	51.60 ± 2.33	72.87 ± 3.89
NAFS	77.83 ± 0.91	89.32 ± 1.71	87.64 ± 0.79	83.86 ± 4.27	54.29 ± 0.69	69.04 ± 1.02	86.07 ± 3.97	83.36 ± 4.24	39.29 ± 0.52	55.12 ± 0.32	73.31 ± 1.12
GDGNN	77.90 ± 1.18	90.12 ± 0.79	87.92 ± 0.36	**83.92 ± 4.36**	54.03 ± 0.34	70.43 ± 1.21	86.12 ± 3.21	83.48 ± 4.05	39.35 ± 0.62	54.83 ± 0.78	73.58 ± 0.96
PairNorm	73.21 ± 1.30	87.86 ± 0.45	86.54 ± 1.24	61.54 ± 3.98	49.05 ± 3.68	63.83 ± 1.98	49.11 ± 5.97	60.12 ± 2.10	26.11 ± 1.97	49.36 ± 2.48	71.37 ± 2.01
GCNII	76.43 ± 1.73	90.06 ± 0.52	**88.62 ± 0.92**	75.75 ± 2.32	39.30 ± 2.41	65.13 ± 2.85	81.77 ± 2.23	76.72 ± 4.90	39.26 ± 0.86	53.83 ± 1.71	72.26 ± 3.86
CPGNN	71.15 ± 0.39	86.81 ± 0.31	84.93 ± 0.70	71.86 ± 1.90	35.06 ± 0.97	57.26 ± 1.82	68.91 ± 2.57	80.16 ± 3.03	35.29 ± 0.36	50.64 ± 2.19	71.01 ± 1.97
GGCN	76.96 ± 1.87	89.35 ± 0.13	86.30 ± 1.01	82.36 ± 3.95	54.26 ± 1.38	70.69 ± 1.39	85.16 ± 2.01	**84.02 ± 5.91**	37.94 ± 1.17	51.69 ± 3.79	71.83 ± 2.20
MixHop	69.03 ± 2.29	86.16 ± 0.69	85.06 ± 1.12	59.32 ± 4.65	33.14 ± 2.21	52.17 ± 3.01	52.93 ± 5.81	60.40 ± 3.79	27.16 ± 1.20	49.05 ± 2.42	69.38 ± 3.11
Geom-GCN	77.92 ± 1.36	89.34 ± 0.50	86.11 ± 1.38	66.16 ± 3.01	40.03 ± 0.71	61.23 ± 1.07	65.27 ± 2.76	62.57 ± 2.86	32.04 ± 1.04	51.77 ± 3.28	71.96 ± 3.26
GPRGNN	76.42 ± 1.05	87.96 ± 1.23	85.02 ± 0.46	79.26 ± 4.76	38.28 ± 1.31	48.20 ± 1.58	83.22 ± 3.62	79.68 ± 7.96	35.77 ± 1.42	53.19 ± 2.80	72.23 ± 3.54
H_2GCN	76.93 ± 1.79	83.61 ± 0.69	87.40 ± 1.67	83.59 ± 5.63	34.89 ± 1.43	63.79 ± 1.15	86.01 ± 5.18	80.74 ± 6.77	36.69 ± 0.67	54.36 ± 3.51	73.06 ± 2.06

Table 4
Accuracy ± stdev on graphs with synthetic data. According to the model, we report the best performance across different layers. The highlighted models stand out as the best for every benchmark. The highlighted gray background represents the second-ranked for every benchmark. △ indicates that the data set adopts the method of subgraph extraction. □ indicates that the data set uses node resampling methods. All benchmarks are based on 10 divided splits (48%:32%:20% nodes per each category for train/validation/test).

Models	Homogeneity			Heterogeneity					
	Citeseer△	Pubmed△	Cora△	Texas□	Squirrel□	Chameleon□	Wisconsin□	Cornell□	Actor□
WSSGCN	**74.62 ± 1.68**	**85.97 ± 0.81**	84.15 ± 1.72	**79.43 ± 5.92**	**52.38 ± 2.41**	**70.01 ± 3.79**	**79.58 ± 6.85**	77.26 ± 7.42	**36.32 ± 2.84**
GCN	68.71 ± 2.89	80.79 ± 1.66	77.31 ± 2.69	50.82 ± 6.32	46.27 ± 2.93	61.77 ± 4.24	47.16 ± 5.46	55.92 ± 6.94	25.04 ± 2.19
GAT	71.13 ± 2.18	82.36 ± 1.97	82.30 ± 2.71	50.03 ± 7.18	38.76 ± 2.83	57.62 ± 3.32	45.98 ± 6.24	60.24 ± 5.46	23.39 ± 3.16
NAFS	73.97 ± 1.28	83.71 ± 1.36	83.92 ± 1.37	73.15 ± 5.12	50.12 ± 1.39	63.71 ± 2.37	76.37 ± 3.36	76.48 ± 4.20	34.10 ± 2.18
GDGNN	74.22 ± 1.36	83.84 ± 1.10	84.02 ± 1.43	74.01 ± 4.37	49.86 ± 1.40	64.12 ± 2.40	76.22 ± 2.93	**77.37 ± 3.96**	35.56 ± 2.33
PairNorm	69.12 ± 1.97	83.28 ± 1.22	82.35 ± 1.96	57.24 ± 5.71	45.39 ± 5.32	60.77 ± 3.24	45.39 ± 4.26	57.45 ± 3.81	22.37 ± 3.20
GCNII	72.19 ± 2.61	83.79 ± 1.26	**84.57 ± 1.46**	72.86 ± 4.69	35.71 ± 3.86	62.98 ± 3.46	76.23 ± 5.40	71.63 ± 6.79	35.43 ± 3.02
CPGNN	66.38 ± 1.71	81.04 ± 0.31	79.28 ± 1.90	67.25 ± 2.63	32.24 ± 1.24	54.36 ± 2.38	62.70 ± 5.96	75.59 ± 5.80	31.43 ± 2.75
Geom-GCN	74.43 ± 2.11	82.93 ± 0.95	83.69 ± 2.71	62.48 ± 5.46	38.25 ± 1.90	55.60 ± 3.29	64.04 ± 4.97	57.35 ± 4.93	29.58 ± 2.73

However, as the dataset size and complexity increase, WSSGCN exhibits a significant performance advantage over other methods. In inhomogeneous datasets, it enhances accuracy by approximately 3%, while in heterogeneous datasets, the improvement amounts to about 5%.

We conduct experiments Table 4 on synthetic data to further verify the generalization ability of our method. The data synthesis methods used for each dataset are as follows:

For the heterogeneous graph datasets (Texas, Wisconsin, Squirrel, Actor, Cornell, Chameleon), we employ subgraph extraction which consists of randomly extracting some nodes and edges from the graph to form subgraphs. As for the homogeneous graph datasets (Cora, Pubmed, Citeseer), we utilize node resampling which involves resampling nodes to alter their quantity and distribution.

4.2.0.2. Performance on width and depth. We also assessed the model's robustness against over-smoothing by evaluating accuracy across a range of layer depths, from 2 to 64 layers. Table 5 presents the outcomes on two heterogeneous and two homogeneous datasets, showcasing the model's behavior. Additionally, we highlight the model with the best performance along with the corresponding optimal depth for each dataset.

As observed from the results in Table 5, PairNorm [59], GCNII [60], GGCN [30], and GPRGNN [62] exhibit notable accuracy fluctuations when layer count increases across different datasets, indicative of over-smoothing. In contrast, WSSGCN attains convergence to optimal performance with just two layers, showcasing its efficacy as a robust baseline for both homogeneous and heterogeneous datasets, even those with intricate structures or substantial data volumes. Notably, WSSGCN can approach optimal performance even on datasets with simpler structures or smaller data volumes, owing to its unique width-based design that achieves convergence in just two layers.

The significance of our proposed model, WSSGCN, lies in its capacity to amalgamate the strengths of both width- and depth-based

Table 5
Accuracy ± stdev on graphs with heterophilic and homophilous features across layers. Gray background highlights second-ranked performance. Benchmarks use 10 splits (48%:32%:20% nodes per category for train/validation/test) from [28].

Models	Chameleon layers							Citeseer layers						
	2	4	8	16	32	64	Best layer	2	4	8	16	32	64	Best layer
WSSGCN	**72.86 ± 2.60**	70.25 ± 1.73	67.33 ± 0.68	67.33 ± 0.68	67.33 ± 0.68	67.33 ± 0.68	**2**	**80.25 ± 1.35**	73.92 ± 1.78	72.56 ± 1.71	72.56 ± 1.71	72.56 ± 1.71	72.56 ± 1.71	**2**
PairNorm	63.83 ± 1.98	60.16 ± 1.23	55.79 ± 1.61	47.24 ± 2.06	47.75 ± 2.37	47.06 ± 2.87	2	73.21 ± 1.30	71.06 ± 1.10	70.69 ± 1.82	56.70 ± 12.43	27.96 ± 11.71	25.41 ± 10.24	2
GCNII	63.74 ± 3.67	65.13 ± 2.85	64.36 ± 1.22	62.71 ± 1.80	60.97 ± 1.91	59.27 ± 1.64	4	72.37 ± 0.98	72.12 ± 0.77	73.86 ± 1.93	75.91 ± 1.02	76.43 ± 1.73	76.26 ± 1.46	32
GGCN	70.06 ± 1.51	70.69 ± 1.39	70.48 ± 1.39	70.57 ± 1.96	70.21 ± 1.10	70.02 ± 0.97	4	76.43 ± 0.97	76.29 ± 1.01	76.96 ± 1.87	76.69 ± 0.76	76.72 ± 0.83	76.55 ± 1.24	10
GPRGNN	48.20 ± 2.08	46.49 ± 3.36	42.02 ± 3.27	40.17 ± 4.11	36.62 ± 5.42	38.37 ± 3.39	2	76.42 ± 1.36	76.37 ± 1.25	76.43 ± 1.68	76.51 ± 1.20	75.36 ± 1.08	73.39 ± 1.73	2
Models	Cora layers							Cornell layers						
	2	4	8	16	32	64	Best layer	2	4	8	16	32	64	Best layer
WSSGCN	88.39 ± 0.48	86.52 ± 1.06	85.27 ± 0.39	85.27 ± 0.39	85.27 ± 0.39	85.27 ± 0.39	2	83.57 ± 5.96	80.21 ± 2.01	78.91 ± 1.32	78.91 ± 1.32	78.91 ± 1.32	78.91 ± 1.32	2
PairNorm	86.54 ± 1.24	85.93 ± 1.09	84.37 ± 1.72	82.68 ± 2.69	62.93 ± 7.96	48.51 ± 6.26	2	52.39 ± 5.26	55.18 ± 6.46	59.36 ± 3.11	59.92 ± 2.44	60.12 ± 2.10	60.12 ± 2.10	32
GCNII	74.81 ± 0.83	85.39 ± 0.75	86.94 ± 1.18	87.26 ± 1.42	88.07 ± 1.53	**88.62 ± 0.92**	**64**	64.54 ± 6.67	67.78 ± 7.06	72.19 ± 5.18	76.72 ± 4.90	73.37 ± 3.71	70.24 ± 2.86	16
GGCN	84.92 ± 1.19	85.22 ± 1.28	85.12 ± 1.56	85.72 ± 1.77	86.30 ± 1.01	85.04 ± 0.62	32	81.13 ± 4.72	82.19 ± 4.67	**84.02 ± 5.91**	83.12 ± 4.37	82.24 ± 4.06	83.37 ± 3.52	**8**
GPRGNN	84.68 ± 0.70	85.02 ± 0.46	84.88 ± 0.96	84.69 ± 0.92	84.11 ± 0.61	83.74 ± 0.72	4	77.15 ± 6.82	78.22 ± 5.41	79.68 ± 7.96	75.43 ± 8.24	73.71 ± 7.21	71.05 ± 5.64	8

Table 6
Efficiency on chosen benchmark: avg. time (ms) per epoch. Model's best performance time used as baseline. Highlighted model is benchmark winner. Gray background highlights second place. Benchmarks based on 10 splits (48%:32%:20% nodes per category for train/validation/test) from [28].

Models	Homogeneity			Heterogeneity							
	Citeseer	Pubmed	Cora	Texas	Squirrel	Chameleon	Wisconsin	Cornell	Actor	Wiki	ogbn-arxiv
WSSGCN	**6.83 ms**	**8.65 ms**	**5.72 ms**	**4.52 ms**	**12.39 ms**	**7.67 ms**	**6.17 ms**	**5.12 ms**	**7.36 ms**	**9.73 ms**	**37.69 ms**
GCN	13.60 ms	15.24 ms	11.26 ms	12.06 ms	30.27 ms	14.46 ms	15.93 ms	14.02 ms	13.38 ms	16.81 ms	326.76 ms
GAT	18.96 ms	23.06 ms	15.62 ms	16.89 ms	49.29 ms	18.06 ms	19.76 ms	13.27 ms	19.94 ms	17.69 ms	352.18 ms
GraphSage	16.49 ms	31.12 ms	14.39 ms	8.16 ms	357.41 ms	65.72 ms	11.36 ms	9.21 ms	28.34 ms	20.35 ms	962.73 ms
GPRGNN	22.37 ms	20.63 ms	20.06 ms	19.33 ms	24.97 ms	16.59 ms	22.83 ms	21.35 ms	20.41 ms	18.26 ms	367.35 ms

models. This enables substantial performance enhancements while preserving model simplicity, demonstrating its versatility across varying complexities and dataset sizes.

4.3. Comparison of training efficiency

We evaluated the training cost of state-of-the-art methods and demonstrated the efficiency of WSSGCN. As shown in Table 6, WSSGCN outperforms the state-of-the-art baseline models regarding training efficiency on both homogeneous and heterogeneous datasets. That is because as the number of training layers increases, the baseline models' parameters increase exponentially, increasing the training cost. On the other hand, WSSGCN only uses a 2-layer deep model, and the increase in width parameters is linear rather than exponential, significantly reducing training time and saving nearly 50% of training time on average.

4.4. Comparison of training time in different sub-stages

We compare the training time of the response-based, feature-based and relationship-based substages as shown in Fig. 3. Training time differences among sub-stages are influenced by several factors:

Complex feature generation: Each sub-stage involves learning to generate feature vectors for nodes, which requires multiple iterations and computations. The Feature-based and Relation-based sub-stages may have more parameters and computations, leading to increased training time. The Relation-based sub-stage, in particular, has the most parameters and consumes the most time.

Hyperparameter tuning: The Relation-based and Feature-based sub-stages may require extensive hyperparameter tuning, especially for the complex models, leading to increased training time. The Relation-based sub-stage, with global optimization, is the most time-consuming process.

4.5. Comparison of convergence time

We conducted comparisons of the algorithm's convergence time and accuracy on isomorphic graph Cora Table 7 and heterogeneous graph Chameleon Table 8, considering different layers.

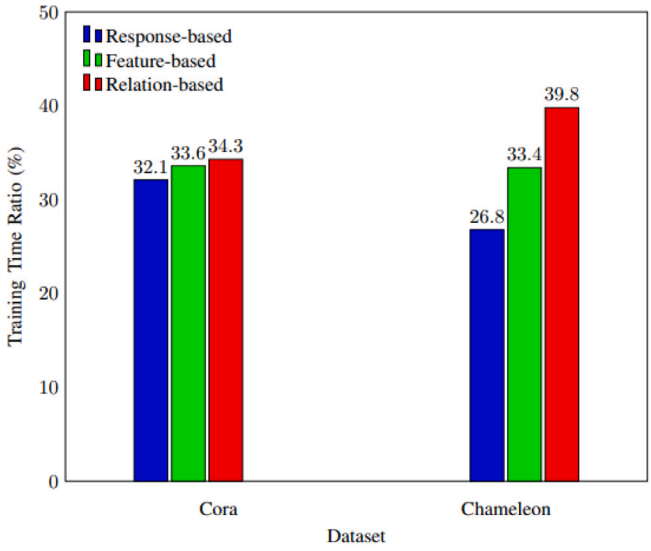


Fig. 3. The proportions of training time at different stages.

Our results show that our algorithm achieves fast convergence and the highest accuracy even with a small number of layers. As the complexity of the dataset increases, we observed that the required number of layers also increases. However, the training time increases linearly, which makes the training cost acceptable. This is mainly due to the linear increase in the width parameter, rather than an exponential increase, which significantly reduces training time.

As the dataset complexity increases, overfitting may occur when the number of layers becomes too large. This is in line with our algorithm's expected behavior, as it is sensitive to the dataset's complexity.

4.6. Ablation study

In this section, we conducted comprehensive ablation studies across diverse datasets to assess the efficacy of our model design. Our approach involved a systematic analysis of WSSGCN's various facets,

Table 7
Model convergence time and training accuracy on Cora dataset.

Model layers	Convergence time	Training accuracy
2	**0.81 h**	**88.39 ± 0.48**
4	1.53 h	86.52 ± 1.06
8	2.96 h	85.27 ± 0.39
16	5.50 h	85.27 ± 0.39
32	10.12 h	85.27 ± 0.39
64	18.30 h	85.27 ± 0.39

Table 8
Model convergence time and training accuracy on Chameleon dataset.

Model layers	Convergence time	Training accuracy
2	**0.23 h**	**72.86 ± 2.60**
4	0.42 h	70.25 ± 1.73
8	0.89 h	67.33 ± 0.68
16	1.57 h	67.33 ± 0.68
32	3.26 h	67.33 ± 0.68
64	6.33 h	67.33 ± 0.68

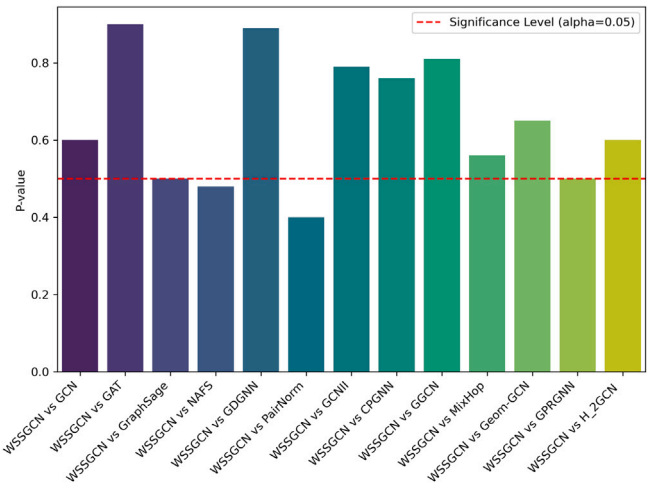


Fig. 4. Statistical comparisons of models.

sequentially removing individual design elements to glean a holistic understanding. The findings are detailed in Table 9.

The experimental results reveal that leveraging sub-stage design methods enhances model performance by 5%–10% on both simple homogeneous and heterogeneous datasets, whereas width-based structures contribute to a 1%–2% improvement. Conversely, for complex homogeneous or heterogeneous datasets, sub-stage design methods yield a performance boost of 2%–5%, with width-based structures demonstrating a more pronounced enhancement of 5%–10%.

The experiments underscore that width-based sub-stage structures yield optimal overall performance. Notably, the influence of sub-stage design methods is particularly pronounced for simpler homogeneous or heterogeneous datasets, whereas width-based structures showcase comparable or marginally superior outcomes. In contrast, the efficacy of width-based structures shines on complex homogeneous or heterogeneous datasets, offering resilience against overfitting—especially valuable when handling expansive data.

To summarize, the strategic amalgamation of sub-stage and width structures within WSSGCN translates to substantial gains in model accuracy, particularly for extensive or diverse datasets.

4.7. Statistical tests

To further validate improvements of our model over other models across multiple datasets, we employed statistical testing methods. Specifically, we utilized the Friedman test and subsequent Nemenyi post-hoc test [64] to ascertain significant performance differences among models across different datasets. This approach not only helps identify situations where our model statistically outperforms others but also provides an objective quantitative method to support the description and comparison of model performance.

As shown in Table 10, our algorithm exhibits a significantly higher sum of ranks and average rank compared to other algorithms across all datasets based on statistical tests. This demonstrates that our algorithm is robust to variations across different datasets and is capable of handling various dataset characteristics. This indicates that our algorithm performs excellently across a wide range of datasets, not only outperforming other algorithms in specific cases but also consistently maintaining high performance across different types and scales of datasets.

According to Fig. 4, the Nemenyi post-hoc test indicates that the critical difference between our algorithm and most other algorithms is around 0.5. Although the degree of difference is not extremely pronounced, it still demonstrates that our algorithm has a clear advantage in performance, showing a certain level of significant difference across multiple datasets. This advantage is statistically validated, indicating that our algorithm performs well in handling different types of datasets.

Table 9
Accuracy ± stdev on graphs with heterophilic and homophilous features for different components. All benchmarks use 10 splits (48%:32%:20% nodes per category for train/validation/test) from [28]. (RE) for response-based, (FE) for feature-based, (RE) for relation-based sub-stages, and (WI) for Width-based. Highlighted green indicates >5% accuracy gain.

Data sets	Ablation				Accuracy ± stdev	Data sets	Ablation				Accuracy ± stdev
	(RE)	(FE)	(RL)	(WI)			(RE)	(FE)	(RL)	(WI)	
Cornell	–	–	–	–	52.71 ± 9.16	Cora	–	–	–	–	51.26 ± 8.73
	✓	–	–	–	70.41 ± 3.25		✓	–	–	–	72.64 ± 4.25
	✓	✓	–	–	78.06 ± 2.79		✓	✓	–	–	79.67 ± 3.16
	✓	✓	✓	–	82.23 ± 4.31		✓	✓	✓	–	86.72 ± 2.01
	✓	✓	✓	✓	**83.57 ± 5.96**		✓	✓	✓	✓	**88.39 ± 0.48**
Citeseer	–	–	–	–	50.15 ± 9.93	Chameleon	–	–	–	–	50.76 ± 7.99
	✓	–	–	–	67.02 ± 7.24		✓	–	–	–	62.16 ± 5.40
	✓	✓	–	–	70.11 ± 6.86		✓	✓	–	–	64.74 ± 5.12
	✓	✓	✓	–	72.96 ± 3.49		✓	✓	✓	–	65.97 ± 2.87
	✓	✓	✓	✓	**80.25 ± 1.35**		✓	✓	✓	✓	**72.86 ± 2.60**

Table 10
Rank and average rank statistics of with heterophilic and homophilous features. Benchmarks are based on 10 splits (48%:32%:20% nodes per category for train/validation/test) from [28].

Models	Homogeneity			Heterogeneity							Average rank
	Citeseer	Pubmed	Cora	Texas	Squirrel	Chameleon	Wisconsin	Cornell	Actor	Wiki	ogbn-arxiv	
WSSGCN	**1**	**1**	2	2	**1**	**1**	**1**	2	**1**	**1**	**1**	**1.27**
GCN	10	7	14	13	2	5	12	12	11	14	13	10.27
GAT	9	13	4	14	7	9	13	10	13	12	11	10.45
GraphSage	11	8	7	4	8	12	7	8	9	9	5	8
NAFS	4	6	5	3	3	4	3	4	3	2	3	3.36
GDGNN	3	2	3	**1**	5	3	2	3	2	3	2	2.63
PairNorm	12	10	8	11	6	7	14	14	14	11	1	10.63
GCNII	7	3	**1**	8	10	6	8	9	4	5	6	6.09
CPGNN	13	11	13	9	12	11	9	6	8	10	12	10.36
GGCN	5	4	9	6	4	2	5	**1**	5	8	9	5.27
MixHop	14	12	11	12	14	13	11	13	12	13	14	12.63
Geom-GCN	2	5	10	10	9	10	10	11	10	7	8	8.36
GPRGNN	8	9	12	7	11	14	6	7	7	6	7	8.54
H_2GCN	6	14	6	5	13	8	4	5	6	4	4	6.81

5. Conclusion

We propose a novel approach called Sub-Stage Structure Perception for constructing WSSGCN, a powerful graph neural network architecture. The proposed approach is conceptually simple yet highly effective in modeling complex graph data. We conduct extensive experiments to investigate the relationship between the types and widths of sub-stage learning processes in the WSSGCN model. We demonstrate the effectiveness of our approach on both homogeneous and heterogeneous datasets, showing that it outperforms state-of-the-art methods in classification tasks.

Furthermore, we analyze the design method based on width and sub-stage structure, which sheds light on the humanized design of the WSSGCN learning process. This analysis allows us to improve the model's performance by optimizing its design. However, as the size of the graph grows, we can anticipate an increase in the complexity of training. Furthermore, to achieve optimal performance, certain hyperparameters involved in WSSGCN might require meticulous tuning, potentially leading to extended training times.

In future work, we plan to explore the application of WSSGCN in weakly supervised learning, which will allow us to improve the learning representation of cross-domain GNN models. That can potentially address many challenging problems in real-world applications, such as recommendation systems, transportation network analysis, and drug discovery.

CRediT authorship contribution statement

Chao Wang: Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Funding acquisition, Formal analysis, Data curation. **Zheng Tang:** Writing – review & editing, Validation, Software, Methodology, Formal analysis, Data curation. **Hailu Xu:** Writing – review & editing, Resources, Methodology, Formal analysis, Data curation.

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 publicly available in a repository: Cora is available at https://doi.org/10.48550/arXiv.1611.08402. Citeseer is available at https://doi.org/10.1145/276675.276685. Pubmed is available at https://doi.org/10.1609/aimag.v29i3.2157. Cornell, Wisconsin, Actor, Texas, Chameleon and Squirrel are available at https://doi.org/10.48550/arXiv.2002.05287. Wiki and ogbn-arxiv are available at https://doi.org/10.48550/arXiv.2206.08583.

References

[1] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014.
[2] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[3] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. Hager, F. Tombari, C. Rupprecht, Semantic image manipulation using scene graphs, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 5212–5221.
[4] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph CNN for learning on point clouds, ACM Trans. Graph. 38 (2018) 1–12.
[5] A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, Protein interface prediction using graph convolutional networks, in: NIPS, 2017.
[6] S. Rhee, S. Seo, S. Kim, Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification, 2017, ArXiv abs/1711.05859.
[7] M. Zitnik, M. Agrawal, J. Leskovec, Modeling polypharmacy side effects with graph convolutional networks, Bioinformatics 34 (2018) i457–i466.
[8] T. Ahmad, S.T.H. Rizvi, N. Kanwal, Transforming spatio-temporal self-attention using action embedding for skeleton-based action recognition, J. Vis. Commun. Image Represent. 95 (2023) 103892.
[9] T. Ahmad, L. Jin, L. Lin, G. Tang, Skeleton-based action recognition using sparse spatio-temporal GCN with edge effective resistance, Neurocomputing 423 (2021) 389–398.
[10] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv: Learn. (2017).
[11] T. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: International Joint Conference on Artificial Intelligence, 2017.
[12] X. Wang, Y. Ye, A.K. Gupta, Zero-shot recognition via semantic embeddings and knowledge graphs, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6857–6866.
[13] K. Marino, R. Salakhutdinov, A.K. Gupta, The more you know: Using knowledge graphs for image classification, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 20–28.
[14] X. Chen, L.-J. Li, L. Fei-Fei, A.K. Gupta, Iterative visual reasoning beyond convolutions, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7239–7248.

[15] T. Wang, R. Wang, D. Jin, D. He, Y. Huang, Powerful graph convolutioal networks with adaptive propagation mechanism for homophily and heterophily, in: AAAI Conference on Artificial Intelligence, 2021.
[16] X. Liang, X. Shen, J. Feng, L. Lin, S. Yan, Semantic object parsing with graph LSTM, 2016, ArXiv abs/1603.07063.
[17] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017.
[18] J. Atwood, D.F. Towsley, Diffusion-convolutional neural networks, in: NIPS, 2015.
[19] A. Jain, A.R. Zamir, S. Savarese, A. Saxena, Structural-RNN: Deep learning on spatio-temporal graphs, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 5308–5317.
[20] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? 2018, ArXiv abs/1810.00826.
[21] G. Li, M. Müller, B. Ghanem, V. Koltun, Training graph neural networks with 1000 layers, 2021, ArXiv abs/2106.07476.
[22] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: AAAI Conference on Artificial Intelligence, 2018.
[23] G. Li, C. Xiong, A.K. Thabet, B. Ghanem, DeeperGCN: All you need to train deeper GCNs, 2020, ArXiv abs/2006.07739.
[24] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, ArXiv abs/1609.02907.
[25] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, in: AAAI Conference on Artificial Intelligence, 2018.
[26] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lió, Y. Bengio, Graph attention networks, 2017, ArXiv abs/1710.10903.
[27] J. Zhu, R.A. Rossi, A.B. Rao, T. Mai, N. Lipka, N. Ahmed, D. Koutra, Graph neural networks with heterophily, in: AAAI Conference on Artificial Intelligence, 2020.
[28] H. Pei, B. Wei, K.C.-C. Chang, Y. Lei, B. Yang, Geom-GCN: Geometric graph convolutional networks, 2020, ArXiv abs/2002.05287.
[29] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P.S. Yu, Y. Ye, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019.
[30] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, D. Koutra, Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks, in: 2022 IEEE International Conference on Data Mining (ICDM), 2021, pp. 1287–1292.
[31] C. Shi, Y. Li, J. Zhang, Y. Sun, P.S. Yu, A survey of heterogeneous information network analysis, IEEE Trans. Knowl. Data Eng. 29 (2015) 17–37.
[32] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. 31 (2017) 833–852.
[33] C. Wang, Z. Tang, The staged knowledge distillation in video classification: Harmonizing student progress by a complementary weakly supervised framework, 2023, ArXiv abs/2307.05201 [Online]. Available: https://api.semanticscholar.org/CorpusID:259766297.
[34] J.I.B. Piaget, The origins of intelligence in children, New York (W W Norton) 1963, 1963, [Online]. Available: https://api.semanticscholar.org/CorpusID:163521533.
[35] J.D. Bransford, A.L. Brown, R.R. Cocking, How people learn: Brain, mind, experience, and school, 1999, [Online]. Available: https://api.semanticscholar.org/CorpusID:143080250.
[36] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (2009) 61–80.
[37] E. Alsentzer, S.G. Finlayson, M.M. Li, M. Zitnik, Subgraph neural networks, 2020, ArXiv abs/2006.10538.
[38] P. Barcel'o, F. Geerts, J.L. Reutter, M. Ryschkov, Graph neural networks with local graph parameters, in: Neural Information Processing Systems, 2021.
[39] M. Zhang, P. Li, Nested graph neural networks, 2021, ArXiv abs/2110.13197.
[40] K. Zhou, Q. Song, X. Huang, D. Zha, N. Zou, X. Hu, Multi-channel graph neural networks, in: International Joint Conference on Artificial Intelligence, 2020.
[41] A. Rahimi, T. Cohn, T. Baldwin, Semi-supervised user geolocation via graph convolutional networks, 2018, ArXiv abs/1804.08049.
[42] Y.C. Ng, R. Silva, Bayesian semi-supervised learning with graph Gaussian processes, in: Neural Information Processing Systems, 2018.
[43] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, 2017, ArXiv abs/1704.01212.
[44] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.
[45] M. Liu, H. Gao, S. Ji, Towards deeper graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020.
[46] Y. Sun, J. Han, Mining heterogeneous information networks: a structural analysis approach, SIGKDD Explor. 14 (2013) 20–28.
[47] S. Zagoruyko, N. Komodakis, Wide residual networks, 2016, ArXiv abs/1605.07146.
[48] J. Tang, J. Sun, C. Wang, Z. Yang, Social influence analysis in large-scale networks, in: Knowledge Discovery and Data Mining, 2009.
[49] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding, 2019, ArXiv abs/1909.13021.
[50] A. Bojchevski, S. Günnemann, Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking, arXiv: Mach. Learn. (2017).
[51] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, 2018, ArXiv abs/1811.05868.
[52] C. Yang, Z. Liu, D. Zhao, M. Sun, E.Y. Chang, Network representation learning with rich text information, in: IJCAI, Vol. 2015, 2015, pp. 2111–2117.
[53] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, Adv. Neural Inf. Process. Syst. 33 (2020) 22118–22133.
[54] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, in: The AI Magazine, 2008.
[55] G. Namata, B. London, L. Getoor, B. Huang, Query-driven active surveying for collective classification, 2012.
[56] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International Conference on Machine Learning, PMLR, 2016, pp. 40–48.
[57] W. Zhang, Z. Sheng, M. Yang, Y. Li, Y. Shen, Z. Yang, B. Cui, NAFS: A simple yet tough-to-beat baseline for graph representation learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 26467–26483.
[58] L. Kong, Y. Chen, M. Zhang, Geodesic graph neural network for efficient graph representation learning, 2022, ArXiv abs/2210.02636 [Online]. Available: https://api.semanticscholar.org/CorpusID:252735241.
[59] L. Zhao, L. Akoglu, PairNorm: Tackling oversmoothing in GNNs, 2019, ArXiv abs/1909.12223.
[60] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, 2020.
[61] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G.V. Steeg, A.G. Galstyan, MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing, 2019, ArXiv abs/1905.00067.
[62] E. Chien, J. Peng, P. Li, O. Milenkovic, Adaptive universal generalized PageRank graph neural network, arXiv: Learn. (2020).
[63] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: Current limitations and effective designs, arXiv: Learn. (2020).
[64] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

Chao Wang received a B.E. degree in Process Equipment and Control Engineering from Jiangnan University in 2007, and an M.S. degree in Computer Application Technology from Northeast University in 2010. In the same year, he joined the China Academy of Railway Sciences and is currently an Assistant Researcher. Since assuming this position in 2012, he has already had multiple invention patents and peer-reviewed publications. His published papers cover interdisciplinary research topics in the fields of neural networks, machine learning, big data and railway signal. His research interests include computer vision, graph neural networks, big data, cloud computing and intelligent rail transit systems.

Chao Wang has led or participated in several key projects in China's rail transportation industry, achieving significant socioeconomic benefits. In 2021, his team won the second prize of the Science and Technology Progress of Beijing Rail Transit Society Award. The project he participated in received the first prize of the China Academy of Railway Sciences Award in 2020. An invention patent he participated in won the China Excellent Patent Award in 2017.

Zheng Tang (M'16) earned his B.Sc. (Eng.) degree with Honors from the joint program by Beijing University of Posts and Telecommunications and Queen Mary University of London in 2014. He further obtained his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Washington in 2016 and 2019, respectively.

In 2019, Dr. Tang joined Amazon as an Applied Scientist for the Amazon One team, serving there until 2021. Currently, he is a Senior Data Science Engineer at NVIDIA's Metropolis division, a position he has held since 2021. He has already had 3 U.S. patents and 17 peer-reviewed publications. His research interests include intelligent transportation systems, object tracking, re-identification, and other topics in the computer vision and machine learning realm.

Dr. Tang has been an Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology since 2021, and an Organizing Committee Member for the AI City Challenge Workshops in conjunction with CVPR since 2020. He will also serve as the Challenge Chair for AVSS 2023, and previously served as an Area Chair for MLSP 2021. He received the Best AE Award of T-CSVT in 2021. A team he led triumphed in the 2nd AI City Challenge Workshop in CVPR 2018, securing the first rank in two tracks. Additionally, his paper was a finalist for two Best Student Paper Awards at ICPR 2016.

Hailu Xu received his Ph.D. degree in Computer Science from Florida International University in the summer 2020. He received B.S. and M.S. degrees in Computer Science from North China Electric Power University, China and The University of Toledo, USA, in 2014 and 2016, respectively. His current research interests include big data system, cloud computing, and operating systems. Currently, he is an assistant professor of the Department of Computer Engineering & Computer Science at California State University, Long Beach, USA.