Layered Division and Global Allocation for Community Detection in Multilayer Network

Fanghao Hu, Junhong Lin, Zhi Cai, and Bang Wang*

***Abstract*—*Community detection in multilayer networks* (CDMN) is to divide a set of entities with multiple relation types into a few disjoint subsets, which has many applications in the Web, transportation, and sociology systems. Recent neural network-based solutions to the CDMN task adopt a kind of *representation fusion and global division* paradigm: Each node is first learned a kind of *layer-wise representations* which are then fused for global community division. However, even with contrastive or attentive fusion mechanisms, the *fused global representations* often lack the discriminative power to capture structural nuances unique to each layer.**

In this paper, we propose a novel paradigm for the CDMN task: Layered Division and Global Allocation (LDGA). The core idea is to first perform layer-wise group division, based on which global community allocation is next performed. Concretely, LDGA employs a multi-head Transformer as the backbone representation encoder, where each head is for encoding node structural characteristics in each network layer. We integrate the Transformer with a community-latent encoder to capture community prototypes in each layer. A shared scorer performs layered division by generating layer-wise soft assignments, while global allocation assigns each node the community label with highest confidence across all layers to form the final consensus partition. We design a loss function that couples differentiable multilayer modularity with a cluster balance regularizer to train our model in an unsupervised manner. Extensive experiments on synthetic and real-world multilayer networks demonstrate that our LDGA outperforms the state-of-the-art competitors in terms of higher detected community modularities. Our code with parameter settings and datasets are available at https://anonymous.4open.science/r/LDGA-552B/.

***Index Terms*—Multi-layer Networks, Graph Learning, Community Detection, Transformer Model.**

I. Introduction

Community detection, the task of identifying groups of nodes that are densely connected and share structural similarities in a network, has long been a central focus of web mining and network science [1]. It supports many applications in diverse domains, like biology, economics, and sociology [2]. Traditional methods, such as modularity maximization [3], label propagation [4], or graph embedding [5], have achieved lots of successes for community detection in *single-layer networks*, where only one type of relation is modeled and all edges are hence of the same type.

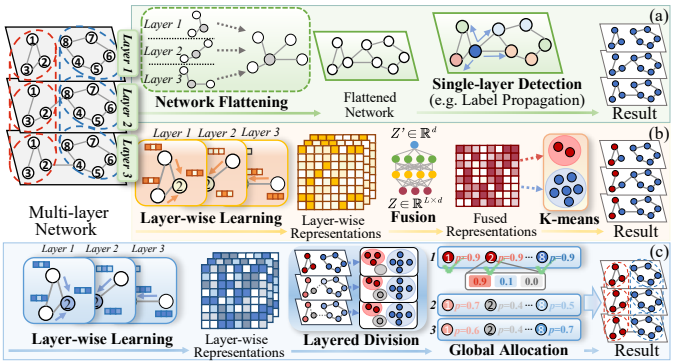

Fig. 1. Illustration of the two paradigms for community detection in multiplex networks. (a) and (b): representation fusion and global division (existing approaches); (c): layered division and global allocations (ours in this paper).

In reality, entities in Web systems and other domains often have multiple yet different types of relations. For example, users may be connected simultaneously through offline friendships, online hyperlinks, or financial transactions. Collapsing these heterogeneous interactions into a single layer discards crucial relational information and risks obscuring the true community structure. *Multilayer networks* address this limitation by preserving all relational types, each by one network layer with edges representing one type of relation. Detecting communities in multilayer networks has numerous applications, such as uncovering the social groups of users on different online platforms [6].

Recently, many neural network-based solutions have been proposed for the community detection in multilayer networks [7]–[13]. However, they all follow a kind of *representation fusion and global division* (**RFGD**) paradigm: neural encoders are designed to first learn layer-wise nodes' representations, that is, a representation is learned in each network layer for each node. Next, *fusion mechanisms* are designed to fuse layer-wise representations into one global representation, that is, a global representation is fused for each node. Finally, clustering algorithms are designed to divide nodes into communities based on computing similarities between their *global representations*.

The RFGD paradigm has evolved significantly. Early strategies, as shown in Fig. 1(a), relied on network flattening , which

Fanghao Hu is with the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan 430074, China (e-mail: hfh@hust.edu.cn).

Junhong Lin is with the Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA (e-mail: junhong@mit.edu).

Zhi Cai is with the School of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: caiz@bjut.edu.cn).

Bang Wang is with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan 430074, China (e-mail: wangbang@hust.edu.cn).

* denotes the corresponding author.

Manuscript received December 1, 2025.

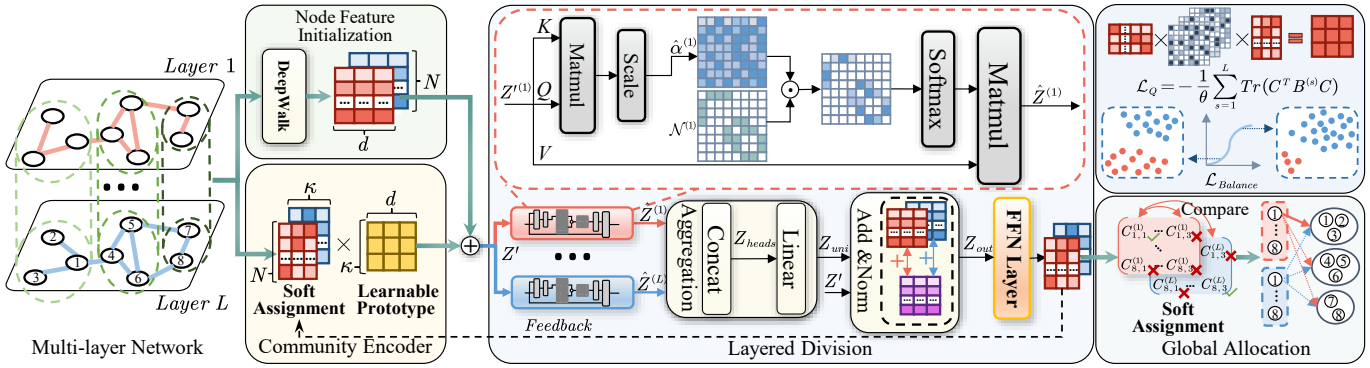

Fig. 2. The overview of LDGA framework (FFN denotes feed-forward networks). LDGA first generates layer-wise soft assignments via Layered Division. In Global Allocation, it assigns each node its highest-confidence community across layers (e.g., node 1 to community 1). The model is trained end-to-end by maximizing a differentiable multilayer modularity objective.

collapses the multilayer structure into a single graph [14], so that standard detection algorithms (e.g., Label Propagation [4], Louvain [15] and Greedy Modularity [3]) can be applied. In contrast, modern neural solutions are illustrated in Figure 1(b). For example, MGCCN [16] evaluates the relevance coefficients of neighboring nodes for learning layer-wise representations; In order to fuse inter-layer semantic information, GEAM [11] first applied a self-attention layer to learn layer-aware representations, then utilized an adaptive fusion mechanism to obtain fused representations. Then, the K-means algorithm is applied to cluster nodes into communities based on the fused representations.

In this paper, we argue that such an RFGD paradigm might not be the best choice for the *community detection in multilayer networks* (CDMN) task. It suffers from three key limitations. (1) **Layer fidelity.** Representation fusion may blur the distinctions of individual layers, either collapsing them into a single view or mixing them with concatenation/additive attention, which weakens the ability to attribute communities to specific relation types. (2) **Representation expressiveness.** Most recent approaches rely on the graph neural networks (e.g., GCN [17]/GAT [18]) for per-layer encoding, but these models are prone to limited expressivity [19], [20], over-smoothing [21], [22], and over-squashing [23]; as a result, they struggle to capture higher-order structural and long-range topological characteristics on each network layer (e.g., cycles [24], [25]). More expressive, attention-based encoders (e.g., transformers [26], [27]) offer a stronger basis for layer-wise learning. (3) **Objective mismatch.** Most methods optimize a kind of objective functions for reconstruction or contrastive surrogates, rather than the actual goal of community detection, leaving representation learning and community division decoupled. Together, these challenges call for a new paradigm that is more layer-faithful and representation-expressive, as well as a more task-aligned training objective for the CDMN task.

To address these challenges, we introduce **L**ayer **D**ivision and **G**lobal **A**llocation framework (**LDGA**) for multilayer network community detection. The core idea is to first perform layer-wise group division, based on which global community allocation is next performed. Take the multilayer network in Figure 1 as an example. For a pre-specified number of $\kappa = 2$ communities, the layered division is to divide nodes into 2 layer-wise communities with probabilities. Node v_2 is assigned to the 1st-community in layer-1 with a probability of 0.9 (2nd-community in layer-1 with probability 0.1), 2nd-community in layer-2 with a probability of 0.4, and 2nd-community in layer-3 with a probability of 0.4. Then the global allocation will assign node v_2 to the global community-1, since it has the highest confidence across all layers.

The LDGA employs a multi-head Transformer integrated with a layer-wise community-latent encoder as the backbone encoder. It first performs **layer-wise division**, where each network layer is encoded by a dedicated Transformer head with sparse dot-product attention restricted to observed edges. A *community-latent encoder* (CLE) augments these per-layer representations with latent community prototypes, controlled by a stability-aware coefficient. An aggregation layer consists of concatenation and down-projection, which integrates the cross-layer signals to capture global structural contexts. A shared feed-forward scorer projects representations into layer-wise assignments. Then the **global allocation** mechanism compares the probabilities and selects labels with the highest confidence to obtain a single consensus partition.

Instead of optimizing a surrogate, LDGA is trained to directly maximize differentiable multilayer modularity, augmented with a balance regularizer that discourages degenerate solutions and behaves consistently across datasets. The model yields a consensus partition across layers by selecting labels with the highest confidence. This end-to-end objective couples attention, feature enrichment, and clustering, aligning learning signals with evaluation metrics.

Experimenting on synthetic mLFR benchmarks and ten real-world multilayer networks spanning social, biological, and transportation domains, LDGA consistently improves community quality, raising NMI/ARI by 17.3%/30.0% on synthetic networks and multilayer modularity by 8.2% on real-world networks, achieving the new state-of-the-art results.

In summary, the contributions of this paper are threefold:

- **Paradigm.** We propose a new LDGA paradigm, which divides nodes at each layer and allocates them to global communities. We implement the LDGA via a transformer with one head per layer as a backbone encoder integrated

with a community encoder.
- **Objective.** We propose a differentiable multilayer modularity with a normalized balance regularizer that directly optimizes the community detection metric and produces a consensus partition.
- **Evaluation.** We conduct extensive evaluations on both synthetic and real-world datasets, and results demonstrate consistent gains over the state-of-the-art competitors.

II. Related Works

This section briefly reviews the related works for community detection in multi-layer networks from two aspects: (1) the strategies employed for fusing information and detecting communities across different network layers, and (2) the learning paradigms used to derive community divisions.

Information Fusion and Community Detection in multi-layer Networks. A central challenge in multi-layer network analysis is integrating information across layers without losing critical structural details. Methodologies have evolved from direct structural aggregation to more nuanced, learning-based approaches. The most direct strategies involve "flattening" or aggregating the multi-layer graph into a single, weighted graph, upon which standard community detection algorithms can be applied [1]. For example, ABACUS [14] sums the adjacency matrices of each layer; Cozzo et al. [28] proposed a method to construct a larger "supra-graph" that represents node-layer pairs. While computationally simple, these methods share a fundamental limitation: the premature compression of information via network flattening inevitably discards the unique topological details of each layer, potentially misleading the global community divisions.

A more sophisticated RFGD paradigm has emerged for the CDMN task with the rise of *graph neural networks* (GNNs) [17]. The general pipeline involves first learning layer-wise embeddings using methods like DeepWalk [29] and subsequently fusing them to produce a unified representation. In [30], a graph convolutional auto-encoder is proposed for learning a low-dimensional vector encoding for each node before clustering. More advanced models use attention mechanisms or network augmentation to learn a weighted combination of layer embeddings, as seen in models like GEAM [11], NFACC [7], and Kazim et al. [31]. However, even these sophisticated methods ultimately merge distinct layer representations into a single global representation for each node. This final fusion step can blur heterogeneous relational signals, losing the strict separation of layer-wise topologies.

Learning Paradigms and Objective Functions for Graph Clustering. Beyond the fusion strategy, the learning objective function profoundly impacts the quality of the detected communities. The majority of recent deep learning methods follow a two-stage pipeline that separates representation learning from the final clustering step. In the first stage, a GNN encoder learns node embeddings in a self-supervised manner, often using contrastive learning frameworks like Deep Graph Infomax (DGI) [32], DMGI [12], and NACC [10]. In the second stage, a standard clustering algorithm like K-means is applied to cluster these embeddings. This decoupled pipeline, however, creates a significant objective mismatch. The GNN is trained to optimize a surrogate objective, such as a contrastive loss, which is not a direct measure of the clustering quality. An embedding optimized for general representation quality is not guaranteed to be optimal for partitioning the graph into dense communities.

To resolve this mismatch, an end-to-end paradigm has emerged that directly optimizes a clustering-specific objective function. For single-layer networks, the recent work [33] introduced a fully differentiable version of the classic modularity metric, allowing a GNN to be trained to directly maximize the clustering quality. Our proposed LDGA framework extends this powerful insight to multi-layer networks. By developing a differentiable multi-layer modularity loss, we align the entire learning process with the goal of community detection in multilayer networks, unifying feature learning and graph clustering into an end-to-end trainable system.

III. Problem Definition

Definition 1 (Multi-layer Network): A multi-layer network is defined as a tuple $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ denotes the shared set of N nodes across all layers. The edge set $\mathcal{E} = \{\mathcal{E}^{(1)}, \mathcal{E}^{(2)}, \ldots, \mathcal{E}^{(L)}\}$ consists of L independent edge sets, where $\mathcal{E}^{(l)}$ represents the intra-layer edges in the l-th layer. No inter-layer edges exist in this structure.

Definition 2 (Multi-layer Network Community Detection): In this work, we focus on disjoint community detection. Formally, disjoint multi-layer network community detection is defined as dividing the node set $\mathcal{V}$ into a partition $\mathcal{C} = \{C_1, C_2, \ldots, C_\kappa\}$ satisfying two conditions:

1) **Coverage:** $\bigcup_{k=1}^{\kappa} C_k = \mathcal{V}$;
2) **Disjointness:** $C_i \cap C_j = \emptyset$ for all $i \neq j$.

IV. Methodology

In this section, we present our LDGA framework, designed to jointly preserve layer fidelity, capture community-level semantics, and directly optimize for clustering quality. Our approach integrates three components: (1) a community-latent encoder (CLE) combined with a layer-aligned sparse transformer for layered division. (2) a global allocation mechanism that selects community labels with highest confidence to produce consensus partition, (3) an end-to-end clustering objective based on differentiable multilayer modularity.

As illustrated in Figure 2, the general workflow is as follows. First, per-layer node embeddings are generated from each graph layer to initialize representations. Next, these embeddings are enriched through the CLE and updated via layer-wise sparse attention, which propagates information only along observed edges within each layer. The outputs are mapped into soft community assignments through a shared scorer that encourages a common semantic space across layers. Finally, the assignments are aggregated into a consensus partition, and the entire model is trained end-to-end by maximizing a differentiable multilayer modularity objective.

A. Node Feature Initialization

Our framework supports multilayer graphs with optional node attributes. If raw node features are provided, they can be directly incorporated as input to each layer. If no attributes are available, which is common in many benchmark datasets, we initialize features via an unsupervised embedding method such as DeepWalk [29], applied independently to each layer. This produces per-layer embeddings $\boldsymbol{Z}^{(s)} = \left[\boldsymbol{z}_1^{(s)}, \boldsymbol{z}_2^{(s)}, \ldots, \boldsymbol{z}_n^{(s)}\right]^T \in \mathbb{R}^{N\times d}$ that capture local connectivity patterns while preserving layer identity. Stacking them yields a feature tensor $\boldsymbol{Z} \in \mathbb{R}^{L\times N\times d}$.

B. Community-Latent Encoder (CLE)

Most existing deep learning methods focus on designing neural network architectures from the perspective of nodes and network layers [11], [34], [35], while overlooking community-level semantics. To enrich representations, we introduce a *community-latent encoder* (CLE), which enriches node features with learnable prototypes that represent latent communities. For each layer s, CLE maintains a learnable prototype matrix $\boldsymbol{E}^{(s)} \in \mathbb{R}^{\kappa\times d}$, where each row vector represents a learnable latent prototype for one community.

At iteration t, CLE augments the node features of layer s using the soft assignments $\boldsymbol{C}^{(s,t-1)} \in \mathbb{R}^{N\times\kappa}$ obtained in the previous iteration (detailed in Section IV-C2). This design follows an expectation–maximization style refinement, where community memberships and prototypes are updated alternately. The enriched features at iteration t are computed as

$$\boldsymbol{Z}'^{(s)} = \boldsymbol{Z}^{(s)} + \eta^{(s)}\boldsymbol{C}^{(s,t-1)}\boldsymbol{E}^{(s)}, \tag{1}$$

where $\eta^{(s)}$ is a learnable fusion coefficient for each layer. At the first iteration ($t = 1$), we set $\boldsymbol{C}^{(s,0)} = 0$ and initialize $\eta^{(s)} = 0$, ensuring that randomly initialized prototypes do not distort the base features $\boldsymbol{Z}^{(s)}$. As training progresses, $\boldsymbol{C}^{(s,t-1)}$ becomes more informative, $\eta^{(s)}$ increases adaptively, allowing CLE to gradually and stably infuse community-level semantics into the node features once both assignments and prototypes have become informative.

C. Layered Division

Following the enrichment of node features by CLE, LDGA applies a *layer-division mechanism* to preserve the unique topology of each layer. Each layer in a multilayer network encodes distinct structural information that should not be prematurely mixed. Therefore, to **preserve layer fidelity**, for a network with L layers, we instantiate a transformer encoder with L attention heads, where each head is exclusively dedicated to processing the information from one layer. This design ensures that information is propagated only within the observed topology of that layer, preventing cross-layer interference during representation learning.

*1) **Layer-aligned Sparse Transformer**:* For a node v_i and its neighbor $v_j \in \mathcal{N}_i^{(s)}$ in layer s, the unnormalized attention score $e_{ij}^{(s)}$ is computed as:

$$\boldsymbol{q}_i^{(s)} = (\boldsymbol{z'}_i^{(s)})^T\boldsymbol{W}_Q^{(s)},\ \boldsymbol{k}_j^{(s)} = (\boldsymbol{z'}_j^{(s)})^T\boldsymbol{W}_K^{(s)},\ e_{ij}^{(s)} = \frac{\boldsymbol{q}_i^{(s)}(\boldsymbol{k}_j^{(s)})^T}{\sqrt{d}} \tag{2}$$

where $\boldsymbol{z'}_i^{(s)}$ denotes the enriched feature of nodes v_i, $\boldsymbol{W}_Q^{(s)}, \boldsymbol{W}_K^{(s)} \in \mathbb{R}^{d\times d}$ are learnable projection matrices in head s. We then normalize these scores using a sparse softmax function defined over the neighborhood $\mathcal{N}_i^{(s)}$:

$$\alpha_{ij}^{(s)} = \frac{\exp(e_{ij}^{(s)})}{\sum_{k\in\mathcal{N}_i^{(s)}}\exp(e_{ik}^{(s)})} \tag{3}$$

This formulation ensures that attention is only computed between directly connected nodes, thereby grounding the model in the observed network structure. Importantly, it also yields computational efficiency: attention complexity becomes proportional to the number of observed edges $\sum_s |\boldsymbol{E}^{(s)}|$ rather than the full N^2 possible pairs.

Through the L attention head, the representation of node v_i in layer s is updated as $\hat{\boldsymbol{z}}_i^{(s)}$ in Equation (4). Collecting all nodes yields the layer-wise output $\hat{\boldsymbol{Z}}^{(s)}$, each capturing the topological structure of its corresponding layer. To integrate these signals, the layer-wise output will pass through an aggregation module consisting of a concatenation and a down-projection. As follows, concatenating across L layers then produces the representations $\boldsymbol{Z}_{\text{heads}} \in \mathbb{R}^{N\times(L\cdot d)}$, where $\|$ denotes the concatenation operator:

$$\hat{\boldsymbol{z}}_i^{(s)} = \mathop{\oplus}_{\forall j\in\mathcal{N}(i)}\left(\boldsymbol{\alpha}_{ij}^{(s)}(\boldsymbol{z'}_j^{(s)})^T\boldsymbol{W}_V^{(s)}\right),\ \boldsymbol{Z}_{\text{heads}} = \mathop{\Vert}_{s\in[1,L]}\left(\hat{\boldsymbol{Z}}^{(s)}\right) \tag{4}$$

The concatenated heads $\boldsymbol{Z}_{\text{heads}}$ are projected through a learnable matrix $\boldsymbol{W}_O \in \mathbb{R}^{(L\cdot d)\times d}$, yielding a unified representation $\boldsymbol{Z}_{uni}$. Finally, this unified representation is element-wise added back to the original representation of each layer via a residual connection.

$$\boldsymbol{Z}_{\text{uni}} = \boldsymbol{Z}_{\text{heads}}\boldsymbol{W}_O, \qquad \boldsymbol{Z}_{\text{out}}^{(s)} = \boldsymbol{Z}_{\text{uni}} + \boldsymbol{Z}'^{(s)} \tag{5}$$

Here, $\boldsymbol{Z}_{\text{uni}}$ aggregates global multi-layer information, while the residual term preserves the layer-wise features from CLE. This design ensures that each layer benefits from cross-layer context without losing its individual structural fidelity.

*2) **Feed-Forward Soft Assignment Scorer**:* After layer-specific attention, LDGA generates layered division probabilities by mapping each layer's output into soft community assignments. For the s-th layer, a shared feed-forward scorer is applied:

$$\boldsymbol{C}^{(s)} = \text{softmax}\left(\boldsymbol{W}_2\left(\sigma(\boldsymbol{W}_1 Z_{\text{out}}^{(s)} + b_1)\right) + b_2\right) \tag{6}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d\times d_{\text{FFN}}}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_{\text{FFN}}\times\kappa}$, $b_1 \in \mathbb{R}^{d_{\text{FFN}}}$, $b_2 \in \mathbb{R}^{\kappa}$, and $\sigma(\cdot)$ denotes a PReLU activation. This produces a soft assignment probability matrix $\boldsymbol{C}^{(s)} \in \mathbb{R}^{N\times\kappa}$, where each row corresponds to a node's distribution over κ communities under layer s. Here, κ is a hyperparameter representing the maximum number of communities pre-specified by the user. The same scorer is shared across all layers, ensuring that assignments from different layers are mapped into a common semantic space. Collectively, these form the layered division outputs $\left\{\boldsymbol{C}^{(s)}\right\}_{s=1}^{L}$.

D. Global Allocation

Given the set of per-layer soft assignments $\left\{\boldsymbol{C}^{(s)}\right\}_{s=1}^{L}$, the global allocation stage integrates the layer-wise assignments into a single, definitive community partition. Intuitively, some layers provide clearer signals than others for a node's membership. To capture the most confident evidence, we aggregate across all layers and assign each node to the community with the highest probability:

$$g_i = \mathop{\text{argmax}}_{s\in[L],p\in[\kappa]}\left(\boldsymbol{C}_{i,p}^{(s)}\right) \tag{7}$$

where $C_{i,p}^{(s)}$ denotes the probability that node v_i belongs to community p in layer s, and $g_i \in \{1,\ldots,\kappa\}$ is the resulting community label. This consensus step ensures that the final partition reflects the strongest structural evidence available, while preserving cross-layer consistency enforced by the shared scorer.

E. Multi-layer Modularity Maximization Loss

A central challenge in multilayer community detection is the design of an objective that aligns directly with the goal of community detection. Many contemporary methods rely on contrastive objectives or supervised proxies, where embeddings are first learned and then clustered by a separate algorithm (e.g., k-means, Louvain). This two-stage process is not end-to-end and does not directly optimize the community structure of interest.

Inspired by Tsitsulin et al. [33], who introduced a differentiable relaxation of the Newman–Girvan modularity, we extend this idea

to multilayer network. We propose a loss function, $\mathcal{L}_{\text{soft}}$, which couples *differentiable multilayer modularity* with a normalized *cluster balance regularizer*:

$$\mathcal{L}_{\text{soft}} = \underbrace{-\frac{1}{\theta}\sum_{s=1}^{L}\text{Tr}\big(\boldsymbol{C}^\top \boldsymbol{B}^{(s)}\boldsymbol{C}\big)}_{\mathcal{L}_Q} + \underbrace{\alpha\cdot\frac{\kappa}{N^2(\kappa-1)}\sum_{p=1}^{\kappa}\Big(\sum_{i=1}^{N}\boldsymbol{C}_{i,p}-\tfrac{N}{\kappa}\Big)^2}_{\mathcal{L}_{\text{Balance}}} \quad (8)$$

Differentiable multilayer modularity ($\mathcal{L}_Q$). Here $\boldsymbol{C}\in\mathbb{R}^{N\times\kappa}$ is the soft assignment matrix output by LDGA, and $\boldsymbol{B}^{(s)} = \boldsymbol{A}^{(s)} - \gamma_s\frac{d^{(s)}(d^{(s)})^\top}{2m_s}$ is the modularity matrix of layer s with degree vector $d^{(s)}$ and edge count m_s. Resolution parameter γ_s controls the scale of communities in layer s. The normalization constant $\theta = \sum_{s=1}^{L} m_s$ represents the total number of edges. This term encourages partitions with dense intra-community edges and sparse inter-community edges consistently across layers. Since modularity is to be maximized, we subtract this term from the loss.

Cluster balance regularizer ($\mathcal{L}_{\text{Balance}}$). Optimizing modularity alone risks degenerate solutions where all nodes collapse into a single cluster. To avoid this, we introduce a balance penalty that discourages extreme imbalance in community sizes. The normalization constant $\frac{N^2(\kappa-1)}{\kappa}$ (derived below) scales this term into $[0,1]$, making the hyperparameter α more interpretable.

1) Derivation of $\mathcal{L}_Q$: The standard modularity Q_m for a multilayer network is defined as:

$$Q_m = \frac{1}{2\omega}\sum_{ijsr}\Bigg[\underbrace{(A_{ijs}-\gamma_s\frac{d_{is}d_{jr}}{2m_s})\delta(s,r)\delta(g_{is},g_{jr})}_{\text{Intra-layer structure}} + \underbrace{\delta(i,j)\ell_{jsr}\delta(g_{is},g_{jr})}_{\text{Inter-layer coupling}}\Bigg]$$

where A_{ijs} is the adjacency of node i to j in layer s and $\delta(g_{is},g_{jr})$ is 1 if nodes belong to the same community, and 0 otherwise. ℓ_{jsr} denotes the coupling parameter. ω is the total multilayer edge strength, defined as $\omega = (\sum_{ijs}A_{ijs} + \sum_{jsr}\ell_{jsr}/2)$. m_s is the total intra-layer edge strength of the sth layer, defined as $m_s = (\sum_{ij}A_{ijs})/2$.

Since our goal is to find a single consensus partition for all layers (i.e., $g_{is} = g_i$ for all s), the inter-layer coupling term becomes redundant. Hence, the objective function simplifies to maximizing the sum of intra-layer modularities:

$$Q_{simple} \propto \sum_{s=1}^{L}\sum_{i,j}\left(A_{ijs}-\gamma_s\frac{d_{is}d_{js}}{2m_s}\right)\delta(g_i,g_j). \quad (9)$$

To enable end-to-end training, we relax the discrete Kronecker delta $\delta(g_i,g_j)$ into continuous soft assignments. Let $\boldsymbol{C}\in\mathbb{R}^{N\times\kappa}$ be the soft assignment matrix. The probability that nodes i and j interact within the same community is approximated by the inner product of their assignment vectors: $\delta(g_i,g_j)\approx(\boldsymbol{C}\boldsymbol{C}^\top)_{ij}$.

Substituting this into the simplified modularity equation and rewriting in matrix trace form:

$$\sum_{s=1}^{L}\sum_{i,j}\boldsymbol{B}^{(s)}_{ij}(\boldsymbol{C}\boldsymbol{C}^\top)_{ij} = \sum_{s=1}^{L}\text{Tr}(\boldsymbol{C}^\top\boldsymbol{B}^{(s)}\boldsymbol{C}). \quad (10)$$

Incorporating the normalization constant θ yields $\mathcal{L}_Q$.

2) Derivation of the Normalization Constant: Consider

$$S = \sum_{p=1}^{\kappa}\Big(\sum_{i=1}^{N}\boldsymbol{C}_{i,p}-\tfrac{N}{\kappa}\Big)^2. \quad (11)$$

The minimum of S is 0 when all clusters are perfectly balanced. Since S is a convex function (sum of squares) constrained by a fixed number N, its maximum is attained at the extreme boundaries of the solution space. Therefore, the maximum occurs when the distribution is most skewed (i.e., all nodes fall into one cluster). In that case:

$$S_{\max} = \Big(N-\tfrac{N}{\kappa}\Big)^2 + (\kappa-1)\Big(0-\tfrac{N}{\kappa}\Big)^2 = \tfrac{N^2(\kappa-1)}{\kappa}. \quad (12)$$

Thus the normalization constant is $\frac{N^2(\kappa-1)}{\kappa}$.

3) Avoiding Trivial Solutions: Our objective naturally rules out the trivial partition:

- **Modularity term:** For a single-cluster solution, $\mathcal{L}_Q = 0$. Any non-trivial partition with positive modularity improves the loss.
- **Balance term:** The trivial solution maximizes $\mathcal{L}_{\text{Balance}}$. Any more balanced partition reduces the penalty.

Together, these effects guarantee that $\mathcal{L}_{\text{soft}}$ prefers non-trivial partitions with positive modularity, driving the model toward meaningful community structures.

By unifying these components, our LDGA framework provides a principled solution to the key challenges of multilayer community detection. Specifically, it ensures that: (1) **layer fidelity is preserved** through the LDGA paradigm, which processes layers independently before deriving a global consensus, thus preventing the premature information fusion of prior methods; (2) **node representations are made more expressive and semantically rich**, as the layer-aligned transformer powerfully captures complex structural patterns while the CLE injects valuable community-level guidance; and (3) **the learning objective is directly aligned with the clustering goal**, as the entire model is trained end-to-end to maximize modularity, thereby resolving the critical objective mismatch inherent in the existing two-stage approaches.

TABLE I
PARAMETER SETTINGS OF MLFR DATASETS

Parameter	Description	Value
d	Average degree	16
maxd	Maximal degree	32
κ	Degree power-law	2
ψ	Community power-law	1

TABLE II
STATISTICS OF REAL-WORLD DATASETS

Datasets	Type	# Nodes	# Edges	Layers
AUCs	Social	61	620	5
CKM	Social	246	1551	3
Lazega	Social	71	2223	3
RM	Social	94	1385	3
Vickers	Social	29	740	3
C.Elegans	Biology	279	5863	3
Drosophila	Biology	8215	43366	7
SACCHPOMB	Biology	4092	63406	7
Kapferer	Finance	39	1018	4
EUAir	Transportation	450	3588	37

V. EXPERIMENTS

A. Experiment Setup

a) Datasets.: To evaluate the performance of our proposed method, we conduct experiments on both synthetic and real-world datasets. Synthetic datasets mainly refer to `mLFR` [36], which is a baseline model for generating multilayer networks. The mixing parameter μ in the mLFR datasets, indicating the probability of edges connecting to the nodes of the other communities, can be adjusted. A smaller μ indicates more obvious community structures, while a larger μ indicates more vague community structures. Table I presents the detailed parameter settings of mLFR datasets in our experiment.

TABLE III
PERFORMANCE ON MLFR SYNTHETIC DATASETS. WE HIGHLIGHT THE **FIRST** AND SECOND BEST RESULTS.

	Metrics	GL	MI	HDMI	DMGI	GEAM	MDLPA	MolTi	NFACC	NACC	LDGA
L=4, μ=0.2	NMI	69.98	69.98	15.45	2.12	**100.0**	69.98	**100.0**	**100.0**	**100.0**	**100.0**
	ARI	50.53	50.53	6.70	1.82	**100.0**	50.53	**100.0**	**100.0**	**100.0**	**100.0**
	Purity	73.63	73.63	66.05	66.05	**100.0**	73.63	**100.0**	**100.0**	**100.0**	**100.0**
	Qm	30.91	30.91	11.63	8.23	**46.40**	30.91	**46.40**	**46.40**	**46.40**	**46.40**
L=4, μ=0.3	NMI	0	2.26	10.72	11.76	32.63	0	87.90	91.83	100.0	**100.0**
	ARI	0	-0.36	8.60	8.31	23.88	0	78.99	93.94	97.95	**100.0**
	Purtiy	63.61	63.61	63.61	63.61	63.61	63.61	90.80	97.19	99.05	**100.0**
	Qm	5.39	5.50	10.77	14.54	25.15	5.39	35.66	35.07	35.66	**35.95**
L=4, μ=0.4	NMI	0	0	2.41	4.91	0.22	0	55.18	2.40	38.93	**88.85**
	ARI	0	0	-1.02	4.85	-0.16	0	42.00	-2.40	31.30	**90.73**
	Purity	65.81	65.81	65.81	65.81	65.81	65.81	77.18	65.81	69.19	**95.84**
	Qm	5.40	5.40	8.29	10.18	21.76	5.40	27.94	21.99	25.44	**28.35**
L=8, μ=0.2	NMI	0	78.82	12.89	8.27	67.21	73.24	**100.0**	**100.0**	**100.0**	**100.0**
	ARI	0	84.97	9.58	7.72	62.98	56.88	**100.0**	**100.0**	**100.0**	**100.0**
	Purity	66.56	93.63	66.56	66.56	82.86	77.61	**100.0**	**100.0**	**100.0**	**100.0**
	Qm	5.60	42.01	10.32	11.24	37.47	33.31	**47.01**	**47.01**	**47.01**	**47.01**
L=8, μ=0.3	NMI	0	10.83	11.52	11.68	22.64	0	89.42	51.58	77.28	**100.0**
	ARI	0	-0.30	5.96	12.27	7.97	0	83.59	38.18	75.28	**100.0**
	Purity	65.92	65.92	65.92	65.92	65.92	65.92	92.99	72.17	88.77	**100.0**
	Qm	5.56	6.11	11.87	12.92	24.28	5.56	36.68	30.90	34.04	**37.59**
L=8, μ=0.4	NMI	0	10.94	0.73	11.42	0.81	0	40.87	1.78	28.01	**52.18**
	ARI	0	0.84	-0.54	8.08	0.51	0	26.50	1.35	18.68	**37.74**
	Purity	65.76	65.76	65.76	65.76	65.76	65.76	71.70	65.76	65.76	**71.88**
	Qm	5.26	7.01	17.72	10.55	24.38	5.26	26.28	24.06	26.37	**28.75**

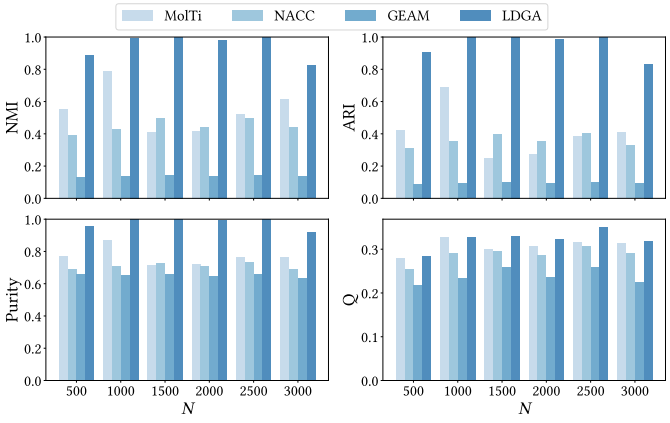

Fig. 3. Experiments with different N on mLFR datasets, N denotes the number of nodes in the network.

We also select ten extensive real-world multilayer networks datasets from domains covering society (AUCs [37], CKM [38], Lazega [39], [40], RM [41], Vickers [42]), biology (C.Elegans [43], Drosophila [44], [45], SACCHPOMB [44], [45]), finance Kapferer [46]), transportation (EUAir [47]). They are all collected manually and without ground-truth labels. Table II presents their basic information.

b) Competitors.: In order to verify the performance of our method, we select nine state-of-the-art community detection methods, which can be divided into the following categories. The first group is traditional community detection algorithms: GenLouvain(GL) [48], M-Infomap(MI) [49], MDLPA [50], MolTi [51]. The second group is five multiplex embedding algorithms: HDMI [13], DMGI [12], GEAM [11], NFACC [7], NACC [10].

c) Evaluation.: We use common evaluation metrics of community detection to evaluate the performance of all the methods, including NMI (Normalized Mutual Information) [52], ARI (Adjusted Rand Index) [53], Purity [54], and multilayer modularity [55]. The maximum value of multilayer modularity is 1, and a higher modularity means a clearer community structure. NMI, ARI, and Purity are only applied to synthetic datasets since they need ground-truth labels, while the modularity of detected communities is applied to both synthetic and real-world datasets. The resolution parameter and coupling parameter of modularity are both set to 1. For all experiments, we run 10 trials with different random seeds, and report the best results in Tables III, IV, and Fig. 3.

d) Parameters: For the initial node feature, we generate a 512-dimensional vector for each network layer. For the DeepWalk, we set the window size as 10, the walks per node as 20 and the walk length as 80. The dimensions of the hidden layers in Transformer and FFN scorer are set to 512 and 1024 separately. We adopt the AdamW optimizer with the initial learning rate 0.0002.

For GL and MI, we use consensus voting to unify the community label. The resolution of GL is set to 1. For the rest of the methods, we follow the parameter settings recommended in their original papers or use the default values provided in their publicly available implementations to ensure a fair comparison.

B. Experimental Results

Q1: How effectively does* LDGA *identify controllable community structures in synthetic networks? On mLFR datasets (Table III), LDGA consistently and significantly outperforms all baseline across different numbers of layers($L = 4, 8$) and mixing parameters ($\mu = 0.2, 0.3, 0.4$). The gains are especially pronounced in challenging settings with high inter-community mixing. For instance, at $L = 4$ and $\mu = 0.4$, LDGA achieves an NMI of 88.85%, whereas the next-best performing method (MolTi), reaches only 55.18%. This shows that LDGA maintains strong discriminative power even when community boundaries are blurred. In contrast, classical multi-layer heuristics (e.g., GL) fail to capture meaningful communities, and deep learning baselines such as DMGI and GEAM lag due to their reliance on surrogate objectives and two-stage clustering pipelines.

Figure 3 presents the scalability of the top-performing methods as the number of nodes (N) increases from 500 to 3000. LDGA not only achieves the best scores overall but also maintains near-perfect NMI, ARI, and Purity as the graph grows, demonstrating superior stability compared to MolTi and NACC, whose performance either fluctuates or declines. This scalability stems from LDGA's sparse attention, whose complexity scales with the number of edges rather than quadratically with nodes, making it well-suited for large-scale multilayer networks.

Q2: How well does* LDGA *generalize across diverse, real-world application domains? LDGA achieves the highest multilayer modularity on all ten real-world datasets (Table IV), demonstrating strong generalizability across domains, including social (CKM, Lazega), biological (C. Elegans, SACCHPOMB), and transportation (EUAir). For instance, on EUAir, LDGA reaches a modularity of 30.37%, compared to 24.71% from the strongest baseline (MolTi). On the large-scale Drosophila network, where several deep learning methods failed due to memory constraints, LDGA not only remained scalable but also achieved the top score of 40.13%. These results confirm that LDGA consistently discovers higher-quality communities and that its end-to-end modularity-driven optimization is effective for uncovering potential community structure in complex, heterogeneous systems.

C. Ablation Study

Q3: What are the contributions of* LDGA*'s core components to its overall performance? The ablation study reported in Table V shows that all core components (Transformer, CLE, and residual connections) are integral and work synergistically to achieve LDGA's success. We systematically removed each component and evaluated the resulting modularity on both synthetic and real-world datasets (e.g., SYN4-0.2 denotes an mLFR network with 500 nodes, 4 layers, and mixing parameter $\mu = 0.2$. (1) **Removing Transformer module ("w/o Trans.") resulted in the most significant performance drop.** For instance, on CKM, the modularity decreased from 61.98% to 56.75%, and on SYN4-0.4, the modularity plummeted from 35.95%

TABLE IV
MODULARITY ON REAL-WORLD DATASETS. WE HIGHLIGHT THE **FIRST** AND SECOND BEST RESULTS.

Network	GL	IM	HDMI	GEAM	DMGI	MDLPA	MolTi	NFACC	NACC	LDGA
AUCs	43.80(7)	42.25(11)	39.54(11)	26.29(10)	23.33(3)	45.36(7)	48.34(5)	47.68(6)	48.28(6)	**48.60(5)**
CKM	61.67(10)	56.55(29)	27.17(6)	53.66(4)	31.14(3)	47.81(46)	61.71(5)	53.23(6)	54.31(6)	**61.98(6)**
Lazega	6.96(1)	6.96(1)	17.65(12)	13.31(8)	08.29(3)	6.96(1)	30.14(3)	29.44(3)	26.12(5)	**32.91(5)**
RM	39.45(9)	32.55(15)	30.26(6)	25.38(3)	25.75(5)	43.22(3)	43.57(5)	38.58(2)	30.32(5)	**45.15(5)**
Vickers	9.89(2)	9.89(2)	27.18(3)	18.45(4)	29.05(3)	22.24(2)	26.63(3)	28.72(3)	24.26(5)	**29.95(5)**
C.Elegans	25.49(8)	33.41(32)	17.97(11)	23.82(9)	15.53(4)	10.93(2)	41.81(7)	41.69(3)	41.42(5)	**42.94(7)**
Drosophila	30.22(139)	33.27(845)	15.44(3)	24.79(17)	18.35(5)	31.25(170)	Overflow.	Overflow.	Overflow.	**40.13(5)**
SACCH.	19.90(62)	18.75(627)	11.16(12)	19.05(9)	12.75(3)	14.38(74)	24.68(30)	6.48(19)	7.04(14)	**28.76(8)**
Kapferer	13.58(6)	20.81(12)	17.37(8)	16.83(9)	20.81(3)	11.26(1)	28.97(4)	24.82(6)	26.19(6)	**29.73(3)**
EUAir	19.12(50)	17.94(175)	23.03(3)	24.15(3)	20.20(3)	18.67(19)	24.71(12)	18.23(2)	15.28(10)	**30.37(6)**

TABLE V
RESULTS OF ABLATION STUDY. WE HIGHLIGHT THE **FIRST** AND SECOND BEST RESULTS.

Method	AUCs	C.Ele.	CKM	Dro.	EUAir	Kap.	Laz.	RM	SAC.	Vic.	SYN4-0.2	SYN4-0.3	SYN4-0.4
w/o Res.	48.33	42.78	59.27	32.33	29.91	**29.73**	**32.91**	42.53	28.48	29.14	**46.40**	35.87	23.27
w/o CLE	48.33	42.61	61.66	36.10	29.66	27.95	**32.91**	44.84	28.36	29.07	**46.40**	35.87	24.04
w/o Trans.	41.03	34.52	56.75	30.39	24.60	25.69	26.98	42.26	17.63	28.66	46.10	31.77	20.06
LDGA	**48.60**	**42.94**	**61.98**	**40.13**	**30.37**	**29.73**	**32.91**	**45.15**	**28.76**	**29.95**	**46.40**	**35.95**	**28.35**

to 31.77%. This highlights the Transformer's crucial role in effectively integrating multilayer information and learning robust cross-layer node representations. (2) **The absence of the CLE ("w/o CLE") also led to a noticeable performance decline.** On Kapferer, the modularity dropped from 29.73% to 27.95%, and on SYN4-0.4, modularity decreased to 24.04%. This suggests that injecting learnable community prototypes enhances the model's ability to discriminate between communities, providing valuable semantic guidance. (3) **The removal of the residual connection caused a slight but consistent decrease in modularity.** For example, on CKM, the modularity reduced to 59.27%, and on Drosophila, it became 32.33%. While less dramatic, this indicates that direct preservation of initial layer-wise features, alongside globally aggregated information, is beneficial for maintaining the quality and stability of learned embeddings.

In summary, these results highlight that each module contributes uniquely, and their combination is essential for LDGA's state-of-the-art performance in multilayer network community detection.

D. Hyperparameter Sensitivity Analysis

Q4: How sensitive is* LDGA *to κ, the pre-specified maximum number of communities, a common challenge in real-world applications where ground truth is unknown? LDGA exhibits strong robustness to the hyperparameter κ, a hyperparameter represents the maximum number of communities. In practice, the true number of communities is rarely known, making robustness to hyperparameter κ critical. Figure 4 plots the modularity achieved by LDGA and four leading deep-learning based baselines (DMGI, GEAM, HDMI, NACC) as κ varies from 2 to 20. For each value of κ, results are averaged over 10 runs with different random seeds, with shaded bands indicating variance.

Across most datasets (e.g., Kapferer, Lazega, RM), LDGA maintains consistently high modularity regardless of κ. In contrast, baselines are highly sensitive: their scores peak sharply at particular values of κ and degrade rapidly elsewhere. This stability highlights a key practical advantage of LDGA—its partitions are guided by the intrinsic multilayer structure rather than overfitting to the pre-specified κ, making it reliable even when the number of communities is misspecified.

Q5: What mechanism underlies* LDGA*'s robustness, and can it automatically determine an appropriate number of communities? LDGA's robustness arises from its ability to discover the most natural partition of a network. The number of non-empty communities it produces quickly converges to a stable, narrow range that is independent of the pre-specified κ.

To investigate this mechanism, we analyzed the relationship between the pre-specified maximum number of communities ("Set κ") and the actual number of non-empty communities produced by the model ("Real κ"). Figure 5 illustrates this relationship on both real-world and synthetic datasets. The results of NACC and NFACC in Drosophila dataset are missing because of memory overflow.

On real-world networks (top row), "Real κ" does not grow unbounded as "Set κ" increases. Instead, it stabilizes within a narrow range. For example, on AUCs, the detected communities consistently converge between 5 and 7, regardless of the chosen "Set κ." This indicates that LDGA captures the inherent community structure rather than forcing arbitrary partitions.

The effect is clearer on mLFR datasets (bottom row), where the ground-truth community number is $\kappa = 3$ (green dashed line). Even when "Set κ" increases well beyond this value, "Real κ" remains tightly centered around 3. This provides strong evidence that the modularity-driven objective effectively prunes redundant or empty clusters, guiding the model toward the most natural partition.

In summary, LDGA not only maintains stable performance across different κ values but also demonstrates an inherent ability to infer an appropriate number of communities—reliably uncovering the latent structure of multilayer networks.

E. Time Analysis

Q6: How is the computational efficiency of* LDGA*? We conducted a time-latency analysis, comparing LDGA's performance against leading deep learning-based baselines. The experiments were performed on ten real-world datasets, measuring the wall-clock time required to complete 100 training epochs. The results(Table VI), demonstrate that LDGA achieves a balance between high performance and computational efficiency.

Our model consistently operates on par with, or faster than, most baseline methods while maintaining high community detection quality. For instance, on the CKM dataset, LDGA's average runtime is 0.0166 seconds per epoch, significantly faster than competitors like GEAM (0.4504s), DMGI (0.0933s), and HDMI (0.0212s). On the

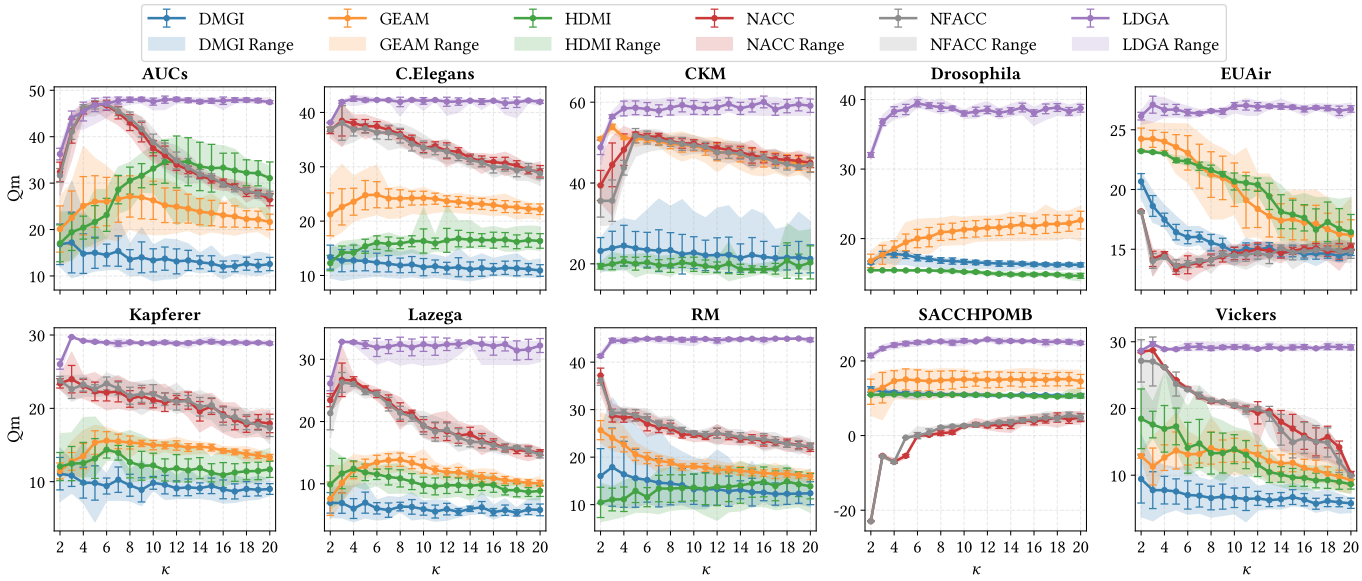

Fig. 4. Modularity with different pre-specified maximum numbers of communities κ on ten real-world datasets.

TABLE VI
RUNNING TIME(S) ON REAL-WORLD DATASETS IN 100 EPOCHS. WE HIGHLIGHT THE **FIRST** AND SECOND MAXIMUM VALUE.

	Method	AUCs	C.Ele.	CKM	Dro.	EUAir	Kap.	Laz.	SAC.	RM	Vic.
AVG Time(in 100 epochs)	DMGI	0.0976	0.0994	0.0933	0.1231	0.9178	**0.1456**	**0.1343**	0.1308	0.0468	**0.1106**
	HDMI	0.0341	0.0222	0.0212	0.1505	0.2115	0.0254	0.0213	0.0901	0.0201	0.0229
	NFACC	0.0118	0.0101	0.0324	Overflow.	0.0710	0.0079	0.0071	1.8646	0.0068	0.0067
	NACC	0.0168	0.0071	0.0069	Overflow.	0.0388	0.0081	0.0058	2.6970	0.0053	0.0057
	GEAM	**0.2160**	**0.5085**	**0.4504**	**71.5126**	**6.5550**	0.1186	0.1232	**18.4292**	**0.1478**	0.0886
	Ours	0.0264	0.0162	0.0166	0.3139	0.2464	0.0195	0.0153	0.1413	0.0135	0.0171
Max Time(in 100 epochs)	DMGI	1.0504	0.2071	0.1877	0.3574	3.1314	0.1934	0.1662	0.4942	0.0747	0.1679
	HDMI	1.3397	0.0442	0.0339	0.4568	0.3320	0.0379	0.0355	0.3212	0.0303	0.0317
	NFACC	0.1571	0.0217	0.0598	Overflow.	0.1967	0.0197	0.0124	3.6519	0.0115	0.0201
	NACC	0.8198	0.0258	0.0174	Overflow.	0.1321	0.0185	0.0133	4.8516	0.0075	0.0190
	GEAM	**2.7211**	**0.8479**	**0.7524**	**94.4320**	**6.7426**	**0.3025**	**0.3098**	**20.3191**	**0.3258**	**0.2758**
	Ours	0.4363	0.0348	0.0360	0.4369	0.3354	0.0457	0.0315	0.1658	0.0277	0.0508
Min Time(in 100 epochs)	DMGI	0.0607	0.0478	0.0462	0.1028	0.4777	0.0576	**0.1086**	0.1137	0.0415	0.0479
	HDMI	0.0267	0.0183	0.0183	0.1457	0.1919	0.0221	0.0178	0.0859	0.0177	0.0182
	NFACC	0.0077	0.0059	0.0122	Overflow.	0.0477	0.0066	0.0051	0.0128	0.0040	0.0040
	NACC	0.0055	0.0045	0.0044	Overflow.	0.0309	0.0051	0.0040	1.5038	0.0041	0.0040
	GEAM	**0.1412**	**0.4026**	**0.3861**	**51.6594**	**6.3588**	**0.0934**	0.0991	**18.0090**	**0.1163**	**0.0567**
	Ours	0.0182	0.0128	0.0120	0.2794	0.2102	0.0152	0.0116	0.1270	0.0113	0.0116
Std(in 100 epochs)	DMGI	0.1770	0.0420	0.0413	0.0176	**0.3911**	0.0454	0.0105	0.0530	0.0055	0.0396
	HDMI	0.0435	0.0031	0.0021	0.0164	0.0189	0.0022	0.0026	0.0150	0.0019	0.0027
	NFACC	0.0155	0.0047	0.0132	Overflow.	0.0330	0.0018	0.0016	**0.8183**	0.0014	0.0021
	NACC	0.0852	0.0028	0.0019	Overflow.	0.0133	0.0024	0.0018	0.7370	0.0007	0.0020
	GEAM	**0.2590**	**0.0986**	**0.0805**	**14.3347**	0.1133	**0.0584**	**0.5708**	0.4689	**0.0565**	**0.0593**
	Ours	0.0412	0.0030	0.0038	0.0151	0.0231	0.0039	0.0032	0.0070	0.0025	0.0078

largest dataset Drosophila, LDGA's running time (0.3139s) remains the same order of magnitude as the fastest method (0.1231s).

The computational advantage of LDGA is primarily attributed to its layer-aligned sparse transformer architecture. As discussed in Section IV-C1, the sparse attention mechanism restricts computations to observed edges only. This design choice avoids the quadratic complexity ($O(N^2)$) associated with standard transformers, making the model's complexity proportional to the number of edges ($O\sum_s |E^{(s)}|$). This ensures that LDGA scales effectively with network size and sparsity, positioning it as a practical and scalable solution for community detection in large, real-world multilayer systems.

VI. CONCLUSION

In this paper, we have proposed LDGA, a novel paradigm for multilayer community detection. The LDGA first performs layered division to generate layer-wise soft assignments, then globally allocates each node to a community with the highest confidence across all layers to form a consensus partition. To achieve better feature expressiveness, we have designed a community-latent encoder into a layer-aligned transformer. A differentiable multilayer modularity loss is designed to align the training process with the community detection task. Experiments on synthetic and real-world networks show consistent improvements in our LDGA over competitors. This work highlights the promise of end-to-end, modularity-driven

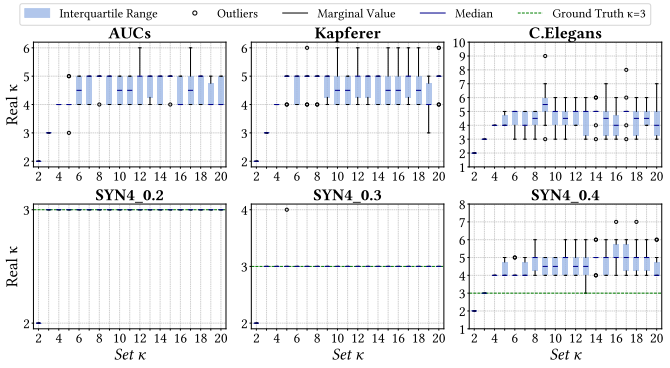

Fig. 5. Real community amount on different datasets

approaches for scalable community detection in Web-scale multilayer systems.

Our framework is still limited in the attention mechanism restricted to observed edges. Future work shall focus on designing a more expressive attention mechanism to capture long-range structural patterns.

REFERENCES

[1] S. Fortunato, "Community detection in graphs," *Physics reports (PR)*, vol. 486, no. 3-5, pp. 75–174, 2010.

[2] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig, "Community detection in networks: A multidisciplinary review," *Journal of Network and Computer Applications (JNCA)*, vol. 108, pp. 87–111, 2018.

[3] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066111, 2004.

[4] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics (PRE)*, vol. 76, no. 3, p. 036106, 2007.

[5] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2016, pp. 855–864.

[6] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks," *Physics reports (PR)*, vol. 544, no. 1, pp. 1–122, 2014.

[7] M. Teng, C. Gao, Z. Wang, and T. Jun, "A multi-layer network community detection method via network feature augmentation and contrastive learning," in *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*. Springer, 2024, pp. 158–169.

[8] H. Song and J. J. Thiagarajan, "Improved deep embeddings for inferencing with multi-layered graphs," in *IEEE international conference on big data (Big Data)*. IEEE, 2019, pp. 5394–5400.

[9] S. Cao, X. Lv, Y. Ma, and X. Ma, "Multi-graph contrastive learning for community detection in multi-layer networks," *IEEE Transactions on Emerging Topics in Computational Intelligence (IEEE TETCI)*, 2024.

[10] M. Teng, Z. Yin, J. Huang, C. Gao, X. Li, V. Nekorkin, and Z. Wang, "Contrastive learning for multi-layer network community detection via learnable network augmentation," *IEEE Transactions on Network Science and Engineering (IEEE TNSE)*, 2025.

[11] B. Wang, X. Cai, M. Xu, and W. Xiang, "A graph-enhanced attention model for community detection in multiplex networks," *Expert Systems with Applications (ESWA)*, vol. 230, p. 120552, 2023.

[12] C. Park, D. Kim, J. Han, and H. Yu, "Unsupervised attributed multiplex network embedding," in *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, vol. 34, no. 04, 2020, pp. 5371–5378.

[13] B. Jing, C. Park, and H. Tong, "Hdmi: High-order deep multiplex infomax," in *Proceedings of the web conference (WWW)*, 2021, pp. 2414–2424.

[14] M. Berlingerio, F. Pinelli, and F. Calabrese, "Abacus: frequent pattern mining-based community discovery in multidimensional networks," *Data Mining and Knowledge Discovery (DMKD)*, vol. 27, no. 3, pp. 294–320, 2013.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment (J STAT MECH-THEORY E)*, vol. 2008, no. 10, p. P10008, 2008.

[16] L. Liu, Z. Kang, J. Ruan, and X. He, "Multilayer graph contrastive clustering network," *Information Sciences (IS)*, vol. 613, pp. 256–267, 2022.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations (ICLR)*, 2019.

[20] B. Egressy, L. Von Niederhäusern, J. Blanuša, E. Altman, R. Wattenhofer, and K. Atasu, "Provably powerful graph neural networks for directed multigraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 38, no. 10, 2024, pp. 11 838–11 846.

[21] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[22] Y. Song, C. Zhou, X. Wang, and Z. Lin, "Ordered GNN: Ordering message passing to deal with heterophily and over-smoothing," in *International Conference on Learning Representations (ICLR)*, 2023.

[23] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," in *International Conference on Learning Representations (ICLR)*, 2020.

[24] Z. Chen, L. Chen, S. Villar, and J. Bruna, "Can graph neural networks count substructures?" *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 10 383–10 395, 2020.

[25] Z. Chen, S. Villar, L. Chen, and J. Bruna, "On the equivalence between graph isomorphism testing and function approximation with gnns," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[26] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[27] J. Kim, S. Oh, and S. Hong, "Transformers generalize deepsets and can be extended to graphs & hypergraphs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 28 016–28 028, 2021.

[28] E. Cozzo, G. F. de Arruda, F. A. Rodrigues, and Y. Moreno, "Multilayer networks: metrics and spectral properties," in *Interconnected networks (IN)*. Springer, 2016, pp. 17–35.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2014, pp. 701–710.

[30] X. Cai and B. Wang, "A graph convolutional fusion model for community detection in multiplex networks," *Data Mining and Knowledge Discovery (DMKD)*, vol. 37, no. 4, pp. 1518–1547, 2023.

[31] M. Kazim, H. Pirim, C. Le, T. Le, and O. P. Yadav, "Multilayer gnn for predictive maintenance and clustering in power grids," *iScience (IS)*, 2025.

[32] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *International Conference on Learning Representations (ICLR)*.

[33] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *Journal of Machine Learning Research (JMLR)*, vol. 24, no. 127, pp. 1–21, 2023.

[34] M. Teng, C. Gao, Z. Wang, and T. Jun, "A multi-layer network community detection method via network feature augmentation and contrastive learning," in *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*. Springer, 2024, pp. 158–169.

[35] Y. Liu, A. Li, A. Zeng, J. Zhou, Y. Fan, and Z. Di, "Motif-based community detection in heterogeneous multilayer networks," *Scientific Reports (JR)*, vol. 14, no. 1, p. 8769, 2024.

[36] P. Bródka, "A method for group extraction and analysis in multilayer social networks," *arXiv preprint arXiv:1612.02377*, 2016.

[37] M. Magnani, B. Micenkova, and L. Rossi, "Combinatorial analysis of multiple networks," *arXiv preprint arXiv:1303.4986*, 2013.

[38] J. Coleman, E. Katz, and H. Menzel, "The diffusion of an innovation among physicians," *Sociometry*, vol. 20, no. 4, pp. 253–270, 1957.

[39] E. Lazega, *The collegial phenomenon: The social mechanisms of cooperation among peers in a corporate law partnership*. OUP Oxford, 2001.

[40] T. A. Snijders, P. E. Pattison, G. L. Robins, and M. S. Handcock, "New specifications for exponential random graph models," *Sociological methodology (SM)*, vol. 36, no. 1, pp. 99–153, 2006.
[41] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Personal and ubiquitous computing (PUC)*, vol. 10, no. 4, pp. 255–268, 2006.
[42] M. Vickers and S. Chan, "Representing classroom social structure," *Victoria Institute of Secondary Education, Melbourne*, 1981.
[43] B. L. Chen, D. H. Hall, and D. B. Chklovskii, "Wiring optimization can relate neuronal structure and function," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 103, no. 12, pp. 4723–4728, 2006.
[44] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, "Biogrid: a general repository for interaction datasets," *Nucleic acids research (NAR)*, vol. 34, no. suppl_1, pp. D535–D539, 2006.
[45] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, "Structural reducibility of multilayer networks," *Nature communications (NC)*, vol. 6, no. 1, p. 6864, 2015.
[46] B. Kapferer, *Strategy and transaction in an African factory: African workers and Indian management in a Zambian town*. Manchester University Press, 1972.
[47] A. Cardillo, J. Gómez-Gardenes, M. Zanin, M. Romance, D. Papo, F. d. Pozo, and S. Boccaletti, "Emergence of network features from multiplexity," *Scientific reports (SR)*, vol. 3, no. 1, p. 1344, 2013.
[48] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *science*, vol. 328, no. 5980, pp. 876–878, 2010.
[49] M. De Domenico, A. Lancichinetti, A. Arenas, and M. Rosvall, "Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems," *Physical Review X*, vol. 5, no. 1, p. 011027, 2015.
[50] O. Boutemine and M. Bouguessa, "Mining community structures in multidimensional networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 4, pp. 1–36, 2017.
[51] G. Didier, A. Valdeolivas, and A. Baudot, "Identifying communities from multiplex biological networks by randomized optimization of modularity," *F1000Research (F1000)*, vol. 7, p. 1042, 2018.
[52] A. Amelio and C. Pizzuti, "Community detection in multidimensional networks," in *IEEE international conference on tools with artificial intelligence (ICTAI)*. IEEE, 2014, pp. 352–359.
[53] V. Gligorijević, Y. Panagakis, and S. Zafeiriou, "Non-negative matrix factorizations for multiplex network analysis," *IEEE transactions on pattern analysis and machine intelligence (IEEE TPAMI)*, vol. 41, no. 4, pp. 928–940, 2018.
[54] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.
[55] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *science (sci.)*, vol. 328, no. 5980, pp. 876–878, 2010.

APPENDIX
ADDITIONAL EXPERIMENT DETAILS

DATASET DETAILS

In this section, we present the detailed description of the synthetic and real-world dataset used in this paper. The synthetic datasets mainly refer to **mLFR** [36], which is a baseline model for generating multilayer networks. We also select ten common multilayer networks to conduct experiment. They are all collected manually and without ground-truth labels.

- **`mLFR`** [36]: We mainly adjust the mixing parameter μ in mLFR datasets, which is the probability of edges connecting to the nodes of the other communities. A smaller μ indicates more obvious community structures, while a larger μ indicates more vague community structures. Table I presents the detailed parameter settings of mLFR datasets in our experiment.

The ten real-world datasets were carefully chosen from domains like society, biology, finance, commerce, and others.

- **`AUCs`** [37]: This multiplex social network contains five types of online and offline relationships (e.g., Facebook, Work, Lunch) among employees at a university's Computer Science department.
- **`C.Elegans`** [43]: This is the connectome of the Caenorhabditis elegans worm, where layers represent different types of synaptic junctions, such as electric and chemical connections.
- **`CKM`** [38]: This dataset contains relationships among physicians in four towns in Illinois, originally collected to study how network ties influenced the adoption of a new drug.
- **`Drosophila`** [44] [45]: This network represents different types of genetic interactions for organisms, sourced from the Biological General Repository for Interaction Datasets (BioGRID).
- **`EUAir`** [47]: This is a transportation network composed of 37 layers, with each layer corresponding to a different airline operating in Europe.
- **`Kapferer`** [46]: This dataset documents interactions in a tailor shop in Zambia, where layers represent two different types of interactions recorded seven months apart.
- **`Lazega`** [39] [40]: This multiplex social network consists of 3 kinds of interactions(Co-work, Friendship and Advice) between partners and associates of a corporate law partnership.
- **`RM`** [41]: This dataset records three social relationships (friendship and proximity at/outside work) between students at the MIT Media Laboratory and MIT Sloan business school.
- **`SACCHPOMB`** [44] [45]: Sourced from the BioGRID repository, this network considers various types of genetic interactions for the fission yeast organism.
- **`Vickers`** [42]: This social network data was collected from 29 seventh-grade students who were asked to nominate classmates based on several different relationships.

BASELINE DETAILS

In this section, we present a detailed description of the selected nine state-of-the-art community detection methods, which can be divided into the following categories. The first group is traditional community detection algorithms:

- **`GenLouvain(GL)`** [48]: A classic method that detects communities by maximizing modularity across layers, treating multilayer structure as a supra-adjacency matrix.
- **`M-Infomap(MI)`** [49]: A method based on the "Map Equation," which identifies communities by minimizing the description length of random walks.
- **`MDLPA`** [50]: A multi-dimensional label propagation algorithm that extends the traditional method by propagating labels alternately between layers to jointly optimize the community partition.

- **MolTi** [51]: This algorithm identifies communities by measuring the consistency between network layers, then performs consensus clustering on the resulting similarity matrices.

The second group is five multiplex embedding algorithms.

- **HDMI** [13]: A high-order deep learning method that captures complex structural features in multilayer networks and combines them with deep transfer learning for community detection.
- **DMGI** [12]: A contrastive learning-based algorithm that extends Deep Graph Infomax to multiplex networks to generate embeddings for community detection and other tasks
- **GEAM** [11]: A graph-enhanced attention model that uses an attention mechanism to integrate node attributes and structural information for generating high-quality node embeddings.
- **NFACC** [7]: A GCN-based method combines network feature augmentation with contrastive learning to integrate intra-layer and inter-layer information for robust multi-layer community detection.
- **NACC** [10]: A self-supervised method fuses intra- and inter-layer features to create a learnable ”anchor” network, which is then used in a contrastive learning framework to train the model.