applied sciences

MDPI

Article

CGAF: Community-Guided Attentive Fusion for Community Detection in Multiplex Networks

Yuchen Liu [1,*] and Hongchang Chen [2]

[1] Institute of Information Technology, Information Engineering University, Zhengzhou 450002, China
[2] National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China; ndscchc@139.com
* Correspondence: liuyuchen2202@163.com

Abstract

Community detection in multiplex networks seeks groups of nodes from several types of relations. Attention mechanisms can integrate these relations, but weights based on embedding compatibility do not explicitly measure the community support provided by a node's neighborhood. We propose Community-Guided Attentive Fusion (CGAF), an end-to-end graph neural network framework that uses neighborhood community evidence to guide relation fusion. Graph contrastive learning encodes refined relation graphs and a learned fused graph to capture relation-specific and shared information. Shared prototypes produce provisional soft memberships, which are used to calculate degree-corrected neighborhood evidence. This evidence adjusts node-specific attention, and a learnable residual coefficient combines the relation embeddings with the fused-graph representation. The prototypes then assign communities to the resulting representations. Joint optimization with contrastive, self-training, and modularity objectives connects representation learning, relation weighting, and community assignment. On ACM, DBLP, and Yelp, CGAF achieves the highest observed means across four community detection metrics among eight comparison methods under the stated protocols. Ablation and parameter analyses show how community evidence, structural supervision, and residual fusion contribute to the learned partitions.

Keywords: multiplex networks; community detection; graph neural networks; attentive fusion; graph contrastive learning; deep clustering

1. Introduction

1.1. Background

Community detection identifies groups of nodes whose connections reveal the organization of a network. Such groups help describe social circles, research areas, and collections of related services or products [1]. In attributed networks, both connectivity and node attributes provide evidence for these groups [2]. Graph neural networks (GNNs) combine the two sources by propagating attributes through graph neighborhoods, making them a natural basis for learning community representations [3,4].

Multiplex networks describe systems in which the same nodes interact through several types of relations. For example, individuals can maintain different connections across social platforms, while researchers can be related through coauthorship and publication venues. Figure 1 illustrates three social-network layers over six individuals. Although the node identities are shared, each layer provides a different neighborhood for the same individual.

Check for updates

Received: 14 September 2026
Revised: 13 October 2026
Accepted: 15 October 2026
Published: 16 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Community detection must therefore integrate the relation layers while retaining the information that distinguishes their contributions [5].

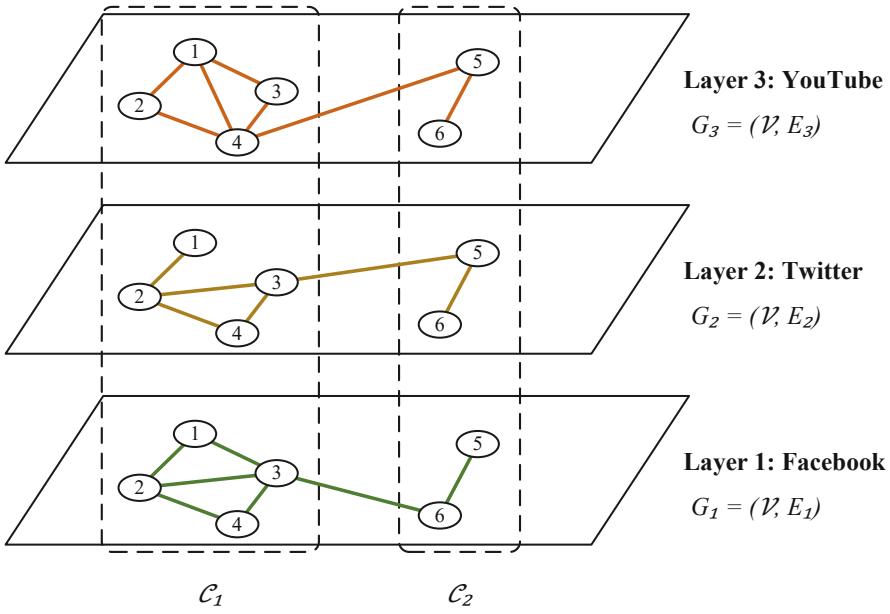


Figure 1. An illustrative multiplex social network with six individuals and three platform layers: Facebook, Twitter, and YouTube. All layers share the node set $\mathcal{V} = \{1, \ldots, 6\}$, while E_1, E_2, E_3 represent platform-specific undirected connections. Dashed boundaries mark the example communities $\mathcal{C}_1 = \{1, 2, 3, 4\}$ and $\mathcal{C}_2 = \{5, 6\}$.

Recent multiplex learning methods improve community representations through graph contrastive learning, adaptive fusion, and graph structure learning. HDMI combines information maximization with attention across relation embeddings [6]. GEAM incorporates graph-level context into node-specific attention and learns communities with a structural objective [7]. InfoMGF learns refined and fused graphs to preserve information shared across relations as well as information specific to individual relations [8].

1.2. Motivation

The usefulness of a relation for community detection depends on the node whose membership is being determined. A relation can provide coherent neighborhood support for one node but connect another node to several communities. A single graph-wide relation weight cannot express this variation. Node-specific attention offers the required flexibility, but scores computed from embedding compatibility do not explicitly measure whether a relation's neighborhood supports a provisional community assignment. The challenge is to make neighborhood community evidence available to the attention mechanism during relation fusion.

Neighborhood support must also be interpreted relative to the distribution of connections in each relation. A community with many incident edges can account for a large fraction of a node's neighbors simply because it occupies a large degree-weighted share of the graph. Using this fraction directly can favor communities with a large share of incident edges. Comparing local soft memberships with a degree-weighted background instead measures how much a neighborhood departs from that baseline. The difference quantifies excess or deficient community support for each node in each relation.

An effective fusion mechanism should use this evidence while preserving the common information captured across layers. We retain a fused-graph representation as an anchor and add a weighted combination of relation embeddings through residual fusion. Community

evidence adjusts the node-specific attention scores that determine these weights. Shared prototypes connect provisional memberships, evidence construction, and final assignments within one differentiable computation. The community objective can therefore influence both a node's representation and the relation weights used to construct it.

1.3. Contributions

We propose Community-Guided Attentive Fusion (CGAF), an end-to-end GNN framework for community detection in attributed multiplex networks. The framework connects neighborhood community evidence, attentive relation fusion, and prototype-based assignment. The main contributions are:

- We formulate degree-corrected neighborhood community evidence from soft assignments and use it to adjust node-specific relation attention. A centered correction makes the evidence contribution zero when the corresponding evidence vector vanishes.
- We integrate the evidence-guided relation representations with a fused-graph anchor through a learnable residual coefficient. Shared prototypes and joint contrastive, self-training, and structural objectives couple attentive fusion with community assignment; inference returns communities directly from the learned membership probabilities.
- We evaluate CGAF against eight methods on ACM, DBLP, and Yelp. It achieves the highest observed mean NMI, ARI, ACC, and macro-F1 under the stated protocols; confidence intervals quantify the differences from the strongest baselines. Six ablation variants and four parameter sweeps examine the proposed components, and a controlled resource profile measures computation and memory requirements as network size increases.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents CGAF, and Section 4 describes the experimental setup. Section 5 reports the comparative results, ablation study, and parameter sensitivity analysis. Section 6 concludes the paper.

2. Related Work

2.1. Community Detection in Multiplex Networks

Multiplex community detection extends single-network clustering by incorporating several relations over a shared node set. Modularity measures within-community connectivity relative to a degree-based null model [9]; its multislice formulation provides a way to couple community structure across network layers [5]. Such structural formulations establish that a layer's observed connections should be assessed against an appropriate background. For attributed multiplex networks, the additional challenge is to combine this structural evidence with the semantic information carried by node features.

Graph filtering and consensus learning provide complementary ways to integrate attributes with multiple relations. MCGC filters node features and learns a consensus graph with a contrastive regularizer [10]. BTGF derives a graph filter from an analysis of the Barlow Twins objective, connecting cross-view correlation with representation learning [11]. BMGC addresses differences in view quality by dynamically identifying a dominant view and using dual signals to guide learning [12]. These approaches account for differences in relation quality when constructing a consensus representation.

Recent GNN methods also address the interaction between graph filtering and community structure. NGCE uses node features to guide contrastive encoding and integrate homophilic and heterophilic information [13]. VGFA combines adaptive graph filtering with variational inference and a cross-layer shared Gaussian mixture model [14]. These methods improve the representations or latent distributions used for clustering. CGAF

instead uses provisional memberships to supply neighborhood evidence to the attention network used for cross-layer fusion.

2.2. Graph Contrastive Learning and Adaptive Fusion

Graph contrastive learning learns representations by relating different observations of the same graph entities. Deep Graph Infomax maximizes information between local node representations and a graph summary [15]. In multiplex settings, DMGI combines relation-specific information maximization with consensus regularization [16], while HDMI incorporates both node–graph and node–attribute information and integrates layer embeddings through attention [6]. These objectives exploit cross-layer dependence without requiring community labels, but the information retained by the encoder still depends on how relations are constructed and combined.

Graph structure learning extends multiplex fusion by adapting the neighborhoods used to compute representations. InfoMGF refines relation graphs and learns a fused graph while preserving both shared and view-specific information [8]. Its contrastive objectives connect refined relations, augmented observations, and the fused representation. CGAF uses these representations to learn relation weights informed by neighborhood community memberships.

Relation attention assigns weights from the information supplied to its scoring network. GEAM combines local and global graph information with node-specific attention and density-based community learning [7]. CGAF supplements embedding compatibility with a degree-corrected neighborhood membership signal. Shared prototypes express this signal in the embedding space, where it provides a centered correction to relation attention.

2.3. Differentiable Community Assignment

Differentiable clustering links the representation space to the partition optimized during training. DEC refines soft memberships using a sharpened auxiliary target [17], and DAEGC combines attentional graph embedding with a self-training clustering objective [18]. DMoN instead optimizes a differentiable modularity objective to recover graph communities [4]. These approaches provide distinct supervisory signals: self-training strengthens confident memberships, whereas modularity rewards structural concentration beyond a degree-based background.

CGAF uses provisional community assignments to construct an explicit input to relation attention. The resulting degree-corrected signal allows community objectives to influence relation weights before the final partition is produced.

3. Materials and Methods

3.1. Problem Formulation and Overview

A multiplex network comprises several relations defined over the same node set. We consider an attributed multiplex network $\mathcal{G} = \{G_v = (\mathcal{V}, A_v, X)\}_{v=1}^{V}$, where $V \geq 2$, $|\mathcal{V}| = N$, $A_v \in \{0,1\}^{N\times N}$ is the symmetric adjacency matrix of relation v, and $X \in \mathbb{R}^{N\times d_x}$ contains shared node attributes. The raw matrices A_v exclude self-loops. Given a prescribed number of communities $K \geq 2$, the objective is to learn a soft assignment matrix $Q \in [0,1]^{N\times K}$ satisfying $Q\mathbf{1}_K = \mathbf{1}_N$. The final partition assigns node i to $\hat{y}_i = \arg\max_k Q_{ik}$. The count K fixes the number of prototype vectors and output columns; it is supplied to the model rather than learned. Training uses attributes, relations, and this count, while individual reference labels are reserved for evaluation. Section 5.5 discusses applications in which K must be estimated.

Figure 2 organizes CGAF into three modules: graph structure refinement, graph contrastive representation learning, and community-guided fusion and assignment. The

first module constructs refined relation graphs and a fused graph. The second uses a shared GCN to encode these graphs and randomly augmented observations, with contrastive objectives linking the resulting representations. The third computes node-specific relation attention, combines the relation embeddings Z_v with the fused-graph embedding Z_f to obtain Z^*, and produces community probabilities Q through a shared prototype head. Self-training and structural objectives jointly optimize this community pathway. Thus, graph fusion determines the adjacency used to obtain Z_f, whereas attentive representation fusion determines Z^*. Table 1 lists the principal quantities.

Graph construction, the shared GCN, random augmentation, and the three contrastive objectives follow InfoMGF [8], with their functional forms unchanged (Sections 3.2 and 3.3). CGAF adds degree-corrected neighborhood evidence, a centered correction to node-specific attention, and residual fusion (Section 3.4). Shared prototypes connect these operations to community assignment; the joint community objectives also update the retained graph learners and encoder.

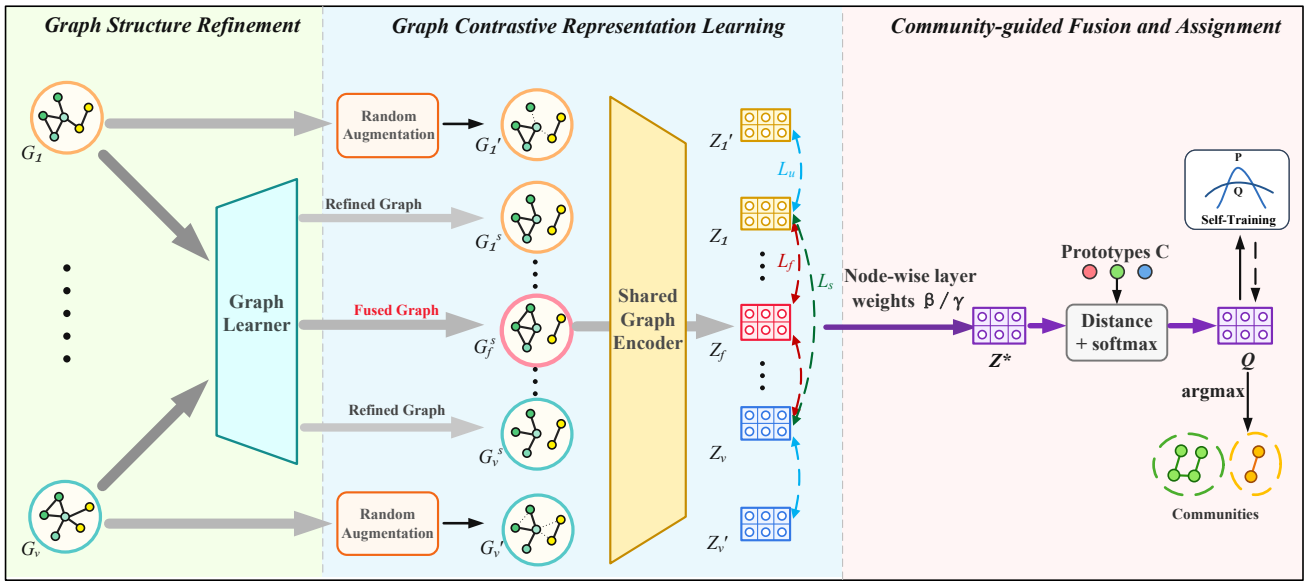


Figure 2. Architecture of CGAF. Refined, augmented, and fused representations are connected through $\mathcal{L}_s$, $\mathcal{L}_u$, and $\mathcal{L}_f$. The fusion module uses provisional memberships $Q^\circ = q_C(Z_f)$ to compute neighborhood evidence and correct node-specific relation attention β. The learnable scalar γ controls the overall contribution of the relation embeddings. Relation embeddings Z_v are mixed with Z_f using γ to obtain Z^*. Shared prototypes yield the final memberships Q, supervised by the sharpened target P and structural objectives. Section 3.4 specifies the evidence and fusion operations grouped within the diagram.

Table 1. Principal notation.

Symbol	Size	Definition
A_v	$N \times N$	Raw adjacency of relation v
X, X^v	$N \times d_x$	Shared attributes and propagated relation features
X_{cat}	$N \times (V+1)d_x$	Concatenated features for fused-graph construction
A_v^s, A_f	$N \times N$	Refined and fused normalized adjacencies
Z_v, Z_f, Z^*	$N \times d$	Relation, fused-graph, and final representations
C	$K \times d$	Shared community prototypes
Q°, Q	$N \times K$	Source and output community assignments
U_v	$N \times K$	Degree-corrected community evidence
β	$N \times V$	Node-specific relation attention
γ	Scalar	Residual mixing coefficient

3.2. Graph Structure Refinement

Relation-specific neighborhoods enter graph construction through fixed feature propagation. Following the simplified graph convolution formulation [19], we compute

$$S_v = \widetilde{D}_v^{-1/2}(A_v + I)\widetilde{D}_v^{-1/2}, \qquad X^v = \mathrm{RowNorm}(S_v^r X), \tag{1}$$

where r is the propagation order, $\widetilde{D}_v = \mathrm{diag}((A_v + I)\mathbf{1}_N)$, and RowNorm normalizes each row to unit Euclidean norm, using a small denominator floor for zero rows. This fixed preprocessing supplies relation-dependent attributes to each graph learner.

Each relation has a two-layer feature-weighting network. Its output is

$$H_v = \mathrm{ReLU}\Big(X^v \odot \mathbf{1}_N(w_{v,1})^\top\Big) \odot \mathbf{1}_N(w_{v,2})^\top, \tag{2}$$

where $w_{v,1}, w_{v,2} \in \mathbb{R}^{d_x}$ are learned feature weights shared across nodes and $\odot$ denotes elementwise multiplication.

For any learner output H, define $R = \mathrm{RowNorm}(H)$ and the cosine-similarity matrix $M = RR^\top$. Let $\mathcal{T}_{k_g+1}(M)$ retain the $k_g + 1$ largest entries in each row, with self-similarity included among the candidates. The graph-construction operator is

$$\mathcal{B}(H) = D_B^{-1/2} B D_B^{-1/2}, \qquad B = \mathrm{ReLU}\left(\frac{\mathcal{T}_{k_g+1}(M) + \mathcal{T}_{k_g+1}(M)^\top}{2}\right), \tag{3}$$

where $D_B = \mathrm{diag}(B\mathbf{1}_N)$ and inverse degrees are stabilized numerically. We obtain $A_v^s = \mathcal{B}(H_v)$. The learned support can differ from the raw support because candidate similarities are computed between all node pairs.

A separate learner constructs the fused graph from $X_{\mathrm{cat}} = [X^1 \| \cdots \| X^V \| X]$, where $\|$ denotes concatenation along the feature dimension. It applies Equation (2) with its own weight vectors of length $(V + 1)d_x$, followed by Equation (3), to obtain A_f. The concatenation is used for graph construction, while the encoder receives the shared attributes X. Thus, the refined and fused graphs are $G_v^s = (\mathcal{V}, A_v^s, X)$ and $G_f^s = (\mathcal{V}, A_f, X)$.

3.3. Graph Contrastive Representation Learning

All learned and augmented graphs use one shared two-layer GCN [3]. Let B denote the adjacency supplied for a forward pass. The encoder applies affine transforms before aggregation:

$$\begin{aligned} H^{(0)} &= \mathrm{Drop}(X), \\ H^{(1)} &= \mathrm{Drop}\Big(\mathrm{ReLU}\Big(B(H^{(0)}W_0 + \mathbf{1}_N b_0^\top)\Big)\Big), \\ f_\theta(X, B) &= \mathrm{RowNorm}\Big(B(H^{(1)}W_1 + \mathbf{1}_N b_1^\top)\Big). \end{aligned} \tag{4}$$

The resulting representations are $Z_v = f_\theta(X, A_v^s)$ and $Z_f = f_\theta(X, A_f)$. During training, dropout is also applied to the normalized learned edge weights. All dropout operations are disabled for inference.

Random augmentation provides additional observations of each original relation. We apply edge dropout to the off-diagonal entries of S_v, preserve its diagonal, and scale retained entries by the inverse retention probability. The augmented graph is $G_v' = (\mathcal{V}, A_v', X)$, with representation $Z_v' = f_\theta(X, A_v')$. Independent feature dropout occurs inside the encoder. Edge-weight dropout is sampled entrywise, so an individual augmented observation can be asymmetric.

The contrastive objective aligns node identity across graph representations. For two projected representation matrices $R, T \in \mathbb{R}^{N \times d_p}$, define $s_{ij} = \cos(R_i, T_j)/\tau_c$ and

$$\mathcal{C}(R,T) = -\frac{1}{2N}\sum_{i=1}^{N}\left[\log\frac{\exp(s_{ii})}{\sum_{j=1}^{N}\exp(s_{ij})} + \log\frac{\exp(s_{ii})}{\sum_{j=1}^{N}\exp(s_{ji})}\right]. \tag{5}$$

The diagonal pairs are positives, and off-diagonal pairs supply negatives. Let p_v^s, p_v^u, and p_v^f be the relation-indexed two-layer projection MLPs used for refined, augmented, and fused representations. Each head has architecture $d \to d_p \to d_p$, with a ReLU between two affine layers. The three contrastive losses are

$$\begin{aligned}
\mathcal{L}_s &= \frac{2}{V(V-1)}\sum_{v<w}\mathcal{C}(p_v^s(Z_v), p_w^s(Z_w)),\\
\mathcal{L}_u &= \frac{1}{V}\sum_{v}\mathcal{C}(p_v^s(Z_v), p_v^u(Z_v')),\\
\mathcal{L}_f &= \frac{1}{V}\sum_{v}\mathcal{C}\Big(p_v^f(Z_f), p_v^s(Z_v)\Big), \qquad \mathcal{L}_{\mathrm{CL}} = \mathcal{L}_s + \mathcal{L}_u + \mathcal{L}_f.
\end{aligned} \tag{6}$$

These objectives respectively align different relations, a relation with its augmented observation, and the fused graph with each relation. These losses act on Z_f, Z_v, and Z_v', while the community assignment module uses the final representation Z^*.

3.4. Community-Guided Fusion and Assignment

3.4.1. Shared Prototypes and Source Assignments

Community evidence requires a common coordinate system across relations. We maintain K learned prototypes $C = [c_1; \ldots; c_K] \in \mathbb{R}^{K \times d}$ and define a Gaussian-distance assignment operator

$$[q_C(z)]_k = \frac{\exp(-\|z - c_k\|_2^2/\tau_q)}{\sum_{j=1}^{K}\exp(-\|z - c_j\|_2^2/\tau_q)}, \qquad Q^\circ = q_C(Z_f). \tag{7}$$

Here, q_C is applied row-wise to a matrix and $\tau_q > 0$ is fixed after initialization. The source matrix Q° describes provisional memberships inferred from the fused representation. The same prototypes are subsequently used for the final assignments, keeping source evidence and output communities in the same coordinate system.

3.4.2. Degree-Corrected Community Evidence

The evidence for relation v is evaluated on its raw adjacency A_v. Write $d^v = A_v\mathbf{1}_N$ and $\omega_v = \mathbf{1}_N^\top d^v$. For a non-isolated node, define

$$b^v = \frac{(d^v)^\top Q^\circ}{\omega_v}, \qquad u_i^v = \frac{(A_v Q^\circ)_i}{d_i^v} - b^v. \tag{8}$$

The first term aggregates the soft community memberships of the node's neighbors. The background b^v averages the same memberships with weights proportional to degree. Their difference measures excess or deficient neighborhood support relative to the distribution encountered at a degree-weighted edge endpoint. This subtraction accounts for the fact that a community associated with many incident edges can receive substantial neighborhood support even without node-specific evidence. For $d_i^v = 0$, we set $u_i^v = 0$; an edgeless relation is assigned zero evidence.

To make the degree correction explicit, let $D_v = \mathrm{diag}(d^v)$. On non-isolated rows, Equation (8) is exactly

$$U_v = D_v^{-1}\left(A_v - \frac{d^v (d^v)^\top}{\omega_v}\right) Q^\circ, \qquad u_i^v = \sum_j \left(\frac{(A_v)_{ij}}{d_i^v} - \frac{d_j^v}{\omega_v}\right) q_j^\circ. \tag{9}$$

Thus, the evidence is a row-normalized modularity residual applied to soft memberships, using the degree-product background of Newman [9]. Division by d_i^v prevents its magnitude from growing merely because the receiving node has more neighbors. The degree-product term defines the reference distribution over edge endpoints. Subtraction removes its predicted contribution: replacing the local neighbor distribution by this background yields zero evidence.

Both $(A_v Q^\circ)_i / d_i^v$ and b^v are probability vectors. Consequently,

$$u_i^v \mathbf{1}_K = 0, \qquad \|u_i^v\|_1 \le 2, \qquad (d^v)^\top U_v = 0. \tag{10}$$

The first two properties follow by subtracting two probability vectors; the last follows from $\mathbf{1}_N^\top A_v = (d^v)^\top$ and includes zeroed isolated rows. The bound is independent of receiving degree, and aggregate evidence has no remaining degree-weighted offset.

Symmetric GCN normalization scales each message by inverse square roots of endpoint degrees [3]; Equation (9) subtracts a community background after row averaging. For fixed topology and a non-isolated receiving node i, changing node j's membership by δq_j changes u_i^v by $((A_v)_{ij}/d_i^v - d_j^v/\omega_v)\delta q_j$. High-degree nodes enter both the observed neighborhood and its background. The resulting signed coefficient expresses their contribution relative to degree-related background support; erroneous hub assignments can still affect multiple receiving nodes. Degree correction therefore addresses background support, while the centered attention operation below ensures a zero correction when this relative evidence vanishes.

The signed evidence vectors measure deviations in community support. We map them to the embedding space using the shared prototypes:

$$t_i = q_i^\circ C, \qquad e_i^v = u_i^v C, \qquad s_i = -\frac{1}{\log K} \sum_{k=1}^{K} q_{ik}^\circ \log q_{ik}^\circ. \tag{11}$$

Here, t_i summarizes the provisional community representation, e_i^v encodes the direction of relation-specific evidence, and s_i measures source-assignment uncertainty. Logarithms are evaluated with a positive numerical floor.

3.4.3. Centered Attention Correction and Residual Fusion

A base attention network first scores the agreement between each relation representation and the fused representation:

$$a_{iv} = g_\phi([z_i^v \| z_i^f]). \tag{12}$$

The function $g_\phi : \mathbb{R}^{2d} \to \mathbb{R}$ is a two-layer MLP with a d-dimensional hidden layer and a hyperbolic-tangent activation. A second MLP $F_\psi : \mathbb{R}^{3d+1} \to \mathbb{R}$, with the same hidden width and activation, supplies a centered evidence correction:

$$\Delta a_{iv} = F_\psi([z_i^v \| t_i \| e_i^v \| s_i]) - F_\psi([z_i^v \| t_i \| 0_d \| s_i]). \tag{13}$$

Centering makes the attention correction depend on the supplied community evidence. Both evaluations share parameters and context, so the correction is exactly zero when the evidence vector is zero. Isolated nodes consequently retain the base attention score.

The corrected scores determine attention across relations:

$$\beta_{iv} = \frac{\exp(a_{iv} + \Delta a_{iv})}{\sum_{w=1}^{V} \exp(a_{iw} + \Delta a_{iw})}, \qquad z_i^* = \mathrm{RowNorm}\left((1-\gamma) z_i^f + \gamma \sum_{v=1}^{V} \beta_{iv} z_i^v \right). \tag{14}$$

The coefficient $\gamma = \mathrm{sigmoid}(\eta)$ controls the overall contribution of relation embeddings, where η is a learned scalar shared by all nodes and relations. Before normalization, Equation (14) forms a convex combination: β_{iv} allocates weights among relations for node i, while γ determines their combined contribution relative to Z_f.

Within each forward pass, Z_f determines Q°, which supplies the evidence used to compute attention and Z^*. Gradients through the source assignments and evidence are retained, allowing the final community objective to update the fused representation and its graph learner.

3.4.4. Community Assignment and Self-Training

The shared prototypes convert the final representation into community probabilities, $Q = q_C(Z^*)$. The logits are negative squared distances from Z^* to C, scaled by τ_q, and the hard partition follows directly from $\arg\max_k Q_{ik}$.

We apply the auxiliary-target self-training principle of DEC [17] while retaining the Gaussian assignment kernel in Equation (7). At each target refresh, we compute

$$f_k = \sum_{i=1}^{N} q_{ik}, \qquad p_{ik} = \frac{q_{ik}^2 / f_k}{\sum_{j=1}^{K} q_{ij}^2 / f_j}, \qquad \mathcal{L}_{\mathrm{ST}} = \frac{1}{N} \sum_{i,k} p_{ik} \log \frac{p_{ik}}{q_{ik}}. \tag{15}$$

The auxiliary target strengthens confident assignments through self-supervision. Squaring the memberships emphasizes stronger predictions, while frequency normalization adjusts for community size. At each refresh, P is computed from the current training-mode output before the optimizer update, detached, and held fixed until the next refresh. The source assignments Q° remain differentiable within every forward pass.

3.4.5. Structural Regularization and Joint Objective

A structural objective links the output communities to the original relations. Using the degree-based modularity background [9], we define the mean soft modularity

$$\mathcal{M}(Q) = \frac{1}{V_+} \sum_{v:\, \omega_v > 0} \frac{1}{\omega_v} \left[\mathrm{tr}(Q^\top A_v Q) - \frac{\| (d^v)^\top Q \|_2^2}{\omega_v} \right], \tag{16}$$

where V_+ is the number of nonempty relations. The value is set to zero if all relations are empty. For one-hot Q, each summand reduces to the usual modularity of that relation. Raw adjacencies are used both here and in Equation (8), so these quantities retain a structural reference while the learned graphs change.

Finally, a soft occupancy regularizer discourages concentration in a small subset of communities:

$$\bar{q} = \frac{1}{N} \sum_i q_i, \qquad \mathcal{L}_{\mathrm{bal}} = K \| \bar{q} \|_2^2 - 1. \tag{17}$$

This term is minimized by uniform average occupancy and therefore introduces a preference for balanced community mass. It does not impose equal hard community sizes.

The complete objective is

$$\mathcal{L} = \lambda_{\mathrm{CL}}\mathcal{L}_{\mathrm{CL}} + \lambda_{\mathrm{ST}}\mathcal{L}_{\mathrm{ST}} - \lambda_{\mathrm{M}}\mathcal{M}(Q) + \lambda_{\mathrm{bal}}\mathcal{L}_{\mathrm{bal}}, \tag{18}$$

where the nonnegative coefficients control the relative strengths of the objectives. The structural term is subtracted because modularity is maximized. Contrastive learning trains the representation pathway, while the community objectives additionally train the prototypes, attention networks, and residual coefficient. Section 4 gives the default coefficients and the parameter-analysis settings.

3.5. Initialization, Optimization, and Inference

Training begins by optimizing the contrastive objective $\mathcal{L}_{\mathrm{CL}}$. We obtain Z_f with dropout disabled and initialize C by running K-means on this representation, using multiple initializations and selecting the solution with the lowest inertia. The Gaussian temperature is fixed to

$$\tau_q = \max\left\{\frac{1}{4}\,\mathrm{median}_{k<j}\,\|c_k - c_j\|_2^2,\ 10^{-4}\right\}. \tag{19}$$

The evidence MLP starts from a neutral correction by initializing its final-layer weights and bias to zero. We initialize the residual coefficient at γ_0 through $\eta = \log(\gamma_0/(1-\gamma_0))$. The initial representation therefore combines the fused-graph embedding with ordinary attentive relation fusion.

Joint training updates the graph learners, shared encoder, projection heads, prototypes, attention networks, and residual coefficient together. Adam is reset at the transition from pretraining, and self-training targets are refreshed at a fixed interval. Algorithm 1 summarizes the computation.

Algorithm 1 Training and inference of CGAF

Require: Raw relations $\{A_v\}_{v=1}^{V}$, shared attributes X, community count K.
Ensure: Soft assignments Q and partition $\hat{y}$.
1: Precompute S_v and X^v using Equation (1).
2: Pretrain graph learners, encoder, and projection heads with $\mathcal{L}_{\mathrm{CL}}$.
3: Initialize C, τ_q, and γ; zero the evidence MLP's output layer.
4: Create a fresh Adam optimizer; initialize the target cache as empty.
5: **for** each joint-training epoch **do**
6: Construct refined, fused, and augmented adjacencies.
7: Encode Z_v, Z_f, and Z'_v using the shared GCN.
8: Obtain Q°; compute U_v, prototype context, and entropy.
9: Compute Δa, β, and Z^*; evaluate $Q = q_C(Z^*)$.
10: **if** the target cache is empty or a refresh is due **then**
11: Set $P \leftarrow \mathrm{stopgrad}(\mathrm{Target}(Q))$.
12: **end if**
13: Evaluate Equation (18) and update all trainable parameters.
14: **end for**
15: Disable dropout; recompute Z_v, Z_f, Z^* and Q from the final parameters.
16: **return** Q and $\hat{y}_i = \arg\max_k Q_{ik}$.

Inference returns the partition in one forward pass with dropout disabled. The final memberships are obtained from the learned prototypes, with K-means confined to their initialization.

3.6. Computational Considerations

Let h and d_p denote the encoder hidden width and projection width, and $J = V(V-1)/2 + 2V$ the number of contrastive comparisons. The following bounds describe a full-

batch training step with dense adjacencies, including backward propagation up to constant factors. Graph construction costs $O(VN^2 d_x) + T_{\text{sel}}$, where T_{sel} denotes row-wise top-k_g selection over $V+1$ graphs. A conservative sorting-based bound is $T_{\text{sel}} = O(VN^2 \log N)$; the actual selection kernel need not sort every row fully. The $2V+1$ encoder passes cost $O(V[N(d_x h + hd) + N^2(h+d)])$. Projections and all-pairs contrastive operations add $O(VN(dd_p + d_p^2) + JN^2 d_p)$. The retained backbone therefore costs

$$T_{\text{back}} = O\Big(VN^2(d_x + h + d) + JN^2 d_p + VN(d_x h + hd + dd_p + d_p^2)\Big) + T_{\text{sel}}. \tag{20}$$

CGAF reuses the backbone embeddings. Its evidence and modularity products, prototype operations, and attention networks add

$$T_{\text{add}} = O(VN^2 K + VNKd + VNd^2). \tag{21}$$

The modularity background uses a degree-vector product without materializing $d^v(d^v)^\top$, and elementwise feature weighting avoids diagonal feature matrices. For T_p pretraining and T_j joint steps, training costs $T_p T_{\text{back}} + T_j(T_{\text{back}} + T_{\text{add}})$, plus one-time propagation, $O(rVN^2 d_x)$, and prototype initialization. With R restarts and I_c Lloyd iterations, initialization has clustering cost $O(RI_c NKd)$ plus a backbone forward pass.

Without checkpointing or tensor offloading, a conservative training-storage bound is

$$O\Big((V+J)N^2 + VN(d_x + h + d + d_p + Kd) + S_\theta\Big), \tag{22}$$

where S_θ accounts for parameters and optimizer state. Quadratic terms include raw and learned graphs and retained contrastive intermediates. Inference removes augmented passes and contrastive matrices, but all-pairs graph construction still requires quadratic storage.

For comparison, a two-layer GCN on one fixed sparse graph with E edges has time $O(N(d_x h + hd) + E(h+d))$ and storage $O(E + N(d_x + h + d) + S_{\text{GCN}})$ [3]. InfoMGF retains Equation (20), and CGAF adds Equation (21). Their quadratic terms arise from learned pairwise graph construction and full-node contrasts. Sparse evidence and modularity multiplication could reduce those operations to $O(K\sum_v |E_v|)$; broader scaling improvements would also require sparse graph learning or sampled contrasts. Section 5.4 measures the resource requirements of the current dense implementation.

4. Experimental Setup

4.1. Datasets

We evaluate CGAF on two academic networks, ACM and DBLP, and one business network, Yelp. The experiments use the processed attributed multiplex graphs distributed with InfoMGF [8]. The datasets differ in node semantics, relation density, and layer count (Table 2).

- **ACM**: Each node represents a paper. Two papers are connected when they share an author (paper–author–paper, PAP) or a subject (paper–subject–paper, PSP). Paper attributes provide textual features, and the reference partition contains three categories.
- **DBLP**: Each node represents an author. The two relations connect authors through coauthored papers (author–paper–author, APA) and common publication venues (author–paper–conference–paper–author, APCPA). The node attributes describe authors, and the reference partition contains four research categories.

- **Yelp**: Each node represents a business. Businesses are connected through shared users (business–user–business, BUB), services (business–service–business, BSB), or rating levels (business–rating level–business, BLB). Business attributes accompany the three relation layers, and the reference partition contains three categories.

Table 2. Dataset statistics. Edges are counted once per undirected pair after removing self-loops.

Dataset	Nodes	Features	Layers	Communities	Edges per Layer
ACM	3025	1902	2	3	13,208/1,098,778
DBLP	2957	334	2	4	1199/730,362
Yelp	2614	82	3	3	262,859/1,237,554/742,346

All methods receive the same node ordering, raw attributes, and binary undirected relation supports. Self-loops are removed from the common supports and introduced where required by each method. CGAF uses two propagation steps and row-wise ℓ_2 normalization; Yelp additionally applies input ℓ_2 normalization as in the backbone configuration. The task is transductive community detection with a prescribed community count. All nodes participate in learning, and reference labels are loaded only by the separate evaluator after final predictions have been saved.

4.2. Comparison Methods

The comparison includes eight methods covering consensus graph learning, variational modeling, information maximization, adaptive fusion, and graph filtering. They include established multiplex baselines and recent graph-clustering approaches.

- MCGC [10]: A multi-view graph-clustering method that combines graph filtering, consensus graph learning, and a contrastive regularizer. It provides a reference based on direct graph optimization.
- VGFA [14]: A variational graph filter autoencoder that combines adaptive filtering with a shared Gaussian mixture model across layers to learn consistent community representations.
- HDMI [6]: A multiplex representation-learning method that maximizes high-order information involving node embeddings, graph summaries, and node attributes, and fuses relation embeddings through attention.
- GEAM [7]: A graph-enhanced attention model for multiplex community detection. We evaluate the authors' released implementation, which trains with a contrastive objective and obtains the partition by K-means.
- BTGF [11]: A multi-relational clustering method that learns a graph filter from an upper-bound analysis of the Barlow Twins objective to exploit correlations across relation views.
- BMGC [12]: A multi-relational graph-clustering method that dynamically mines a dominant view and uses dual supervisory signals to address differences in view quality.
- NGCE [13]: A multi-view graph-clustering method that uses node features to guide contrastive encoding and combines information associated with homophilic and heterophilic graph patterns.
- InfoMGF [8]: An information-aware multiplex graph-fusion method that learns refined relation graphs and a fused graph through contrastive objectives. We use its random-augmentation configuration and apply K-means to the final fused representation.

All methods are evaluated on the same dataset releases, with common node ordering, raw attributes, relation supports, and reference labels. Table 3 summarizes the implementation choices for four comparison methods. Method-specific preprocessing and optimization

settings are described below. Training and checkpoint selection use fixed budgets or loss-based stopping. Published scores reflect the input and evaluation protocols of the original studies; rerunning the methods on these shared inputs provides a consistent basis for comparison.

Table 3. Input, implementation, and selection choices for four comparison methods.

Method	Implementation Choice	Purpose
GEAM	Released model, contrastive objective, and K-means readout; shared attributes X replace per-layer DeepWalk features	Evaluate the authors' implementation with the common attributed inputs
MCGC	Separate feature filtering for each view; full first-view adjacency for initialization; fixed parameters and final consensus updates	Preserve relation-specific inputs and use a fixed evaluation protocol
BMGC	Include all K pseudo-clusters; move a point from a nonsingleton group to each empty cluster before updating centers; use the final model	Define a finite center for every prescribed cluster
NGCE	Use current fused predictions for contrastive grouping; fix the budget at $T_0 + 1000$; scale Yelp attributes to $[0, 1]$	Keep groups aligned with updated memberships and satisfy the binary-cross-entropy target range

For GEAM, we use the released code at commit 1ac5d5d (https://github.com/HustMinsLab/GEAM/tree/1ac5d5d36358c9c1df001f9bc15e53b7546c2b15, accessed on 14 September 2026), retaining its model, optimized contrastive loss, graph preprocessing, and K-means readout. The common node attributes are supplied through its feature-input interface in place of the original DeepWalk vectors. Computation runs on the GPU, with K-means on the CPU. We use the default hidden width of 512 and learning rate of 0.001. The reported partition is the last K-means output of the released loop, obtained after 100 parameter updates with 20 clustering restarts. All five runs contribute to the reported means and sample standard deviations.

4.3. Implementation Details

CGAF uses a shared two-layer GCN and two training stages with 100 epochs each. Both stages use Adam [20] with zero weight decay. The dropout probability is 0.5 for features and graph weights, the contrastive temperature is $\tau_c = 0.2$, and the self-training target is refreshed every five epochs. The main configuration sets $(\lambda_{\mathrm{CL}}, \lambda_{\mathrm{ST}}, \lambda_{\mathrm{M}}, \lambda_{\mathrm{bal}}) = (1, 0.5, 0.1, 0.01)$ and initializes the learnable residual coefficient at $\gamma_0 = 0.1$. Table 4 gives the dataset-specific settings.

Table 4. CGAF architecture and optimization settings.

Parameter	ACM	DBLP	Yelp
GCN hidden width	128	64	128
Embedding/projection dimension	64/64	32/32	64/64
Graph-neighbor parameter k_g	15	10	15
Propagation order/GCN depth	2/2	2/2	2/2
Attention hidden width	64	32	64
Pretraining/joint-training epochs	100/100	100/100	100/100
Pretraining learning rate	10^{-2}	10^{-2}	10^{-3}
Joint-training learning rate	10^{-3}	10^{-3}	10^{-3}

The main comparison uses five independent random seeds, 21–25, for each method and dataset. CGAF initializes prototypes from the pretrained fused representation using K-means with 20 restarts, and representation-based baselines use the same number of

restarts for their final clustering. For controlled parameter continuations, settings share an initialized model and reset the optimizer and random states. Training uses Python 3.9.7 and PyTorch 2.4.0 with CUDA 12.4 on an NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB memory; CPU-based methods use four threads. MCGC, VGFA, BTGF, and BMGC run on the CPU; the other methods run on the GPU. Table 5 lists the baseline optimization budgets.

Table 5. Optimization budgets of the comparison methods.

Method	Optimization Budget
MCGC	Three outer iterations, each with 100 updates
VGFA	100 pretraining epochs and 200 main-training epochs
HDMI	Loss-based stopping with patience 100; at most 10,000 epochs
GEAM	Released 101-iteration loop; final K-means readout after 100 parameter updates
BTGF	400 epochs
BMGC	400 epochs
NGCE	$T_0 + 1000$ epochs, with $T_0 = 650$ on DBLP and 800 otherwise
InfoMGF	100 epochs with random augmentation

4.4. Evaluation Metrics

We measure agreement with the reference communities using NMI, ARI, ACC, and macro-F1. NMI uses arithmetic normalization, $2I(Y;\hat{Y})/(H(Y)+H(\hat{Y}))$, and ARI adjusts pairwise clustering agreement for chance. ACC is defined as

$$\mathrm{ACC} = \frac{1}{N}\max_{\pi}\sum_{i=1}^{N}\mathbb{I}[y_i = \pi(\hat{y}_i)], \tag{23}$$

where π is a one-to-one mapping obtained by the Hungarian algorithm. The same mapping is used to compute macro-F1, which assigns equal importance to each reference category.

The main results report the mean and sample standard deviation over five runs, expressed in percent. Differences between methods are computed from unrounded means and expressed in percentage points. The ablation comparison reports mean and sample standard deviation over five repetitions for NMI and ARI. Parameter analysis uses five additional seeds, 31–35, and pairs each parameter setting with the default configuration from the same initialization. Structural modularity, assignment entropy, and relation attention are retained as diagnostics for interpreting the sensitivity curves.

For the twelve main comparisons, we report descriptive two-sided 95% confidence intervals for the difference between CGAF and the baseline with the highest observed mean in each dataset–metric combination. On ACM and DBLP, the selected comparators (NGCE or BMGC) are separately initialized algorithms. We use Welch intervals, $(\bar{x}-\bar{y}) \pm t_{0.975,\nu}\sqrt{s_x^2/5 + s_y^2/5}$, with sample variances and Welch–Satterthwaite degrees of freedom. On Yelp, the comparator is InfoMGF-RA, whose saved pretrained backbone is also the starting point for CGAF in each run. We therefore use paired intervals $\bar{\delta} \pm t_{0.975,4}s_\delta/\sqrt{5}$ over these five shared-pretraining differences. These approximate intervals describe variation across initializations on each fixed graph. They are pointwise intervals, without adjustment for the twelve comparisons or selection of the strongest baseline.

Two additional diagnostics describe the partitions in the parameter analysis. The mean normalized assignment entropy is $-\frac{1}{N\log K}\sum_{i,k} Q_{ik}\log Q_{ik}$, evaluated on the final output Q. Hard-partition modularity is Equation (16) evaluated with the one-hot assignments induced by $\hat{y}_i$ and averaged over nonempty relations.

5. Results and Analysis

5.1. Comparison with Existing Methods

CGAF has the highest observed mean in all twelve dataset–metric combinations in Table 6 under the protocols in Section 4. Table 7 complements these rankings with confidence intervals for the differences from the strongest baseline in each comparison.

Table 6. Community detection performance on ACM, DBLP, and Yelp (mean ± sample standard deviation over five seeds, in percent). Higher values are better. Bold and underlined values indicate the highest and second-highest means, not statistical significance. F1 denotes macro-F1; GEAM uses the authors' released implementation with shared node attributes, and InfoMGF uses random augmentation.

	ACM				DBLP				Yelp			
Method	**NMI**	**ARI**	**ACC**	**F1**	**NMI**	**ARI**	**ACC**	**F1**	**NMI**	**ARI**	**ACC**	**F1**
MCGC	44.50 ± 0.00	36.86 ± 0.00	61.75 ± 0.00	52.19 ± 0.00	0.45 ± 0.00	0.07 ± 0.00	28.93 ± 0.02	23.06 ± 0.02	13.53 ± 0.03	3.31 ± 0.02	46.87 ± 0.06	39.77 ± 0.05
VGFA	33.84 ± 2.63	21.94 ± 3.45	61.34 ± 2.19	62.85 ± 2.33	61.97 ± 3.40	66.08 ± 3.05	84.94 ± 1.29	84.18 ± 1.20	33.31 ± 8.72	31.18 ± 11.37	61.60 ± 8.24	53.34 ± 1.89
HDMI	65.64 ± 1.13	66.75 ± 1.28	86.78 ± 0.66	86.50 ± 0.70	68.99 ± 3.08	74.29 ± 3.47	88.77 ± 1.80	87.99 ± 1.95	39.02 ± 0.00	42.53 ± 0.00	65.07 ± 0.00	56.74 ± 0.00
GEAM	63.55 ± 0.84	66.57 ± 1.20	87.03 ± 0.62	87.13 ± 0.64	33.16 ± 12.66	30.31 ± 13.82	53.66 ± 11.37	47.95 ± 11.84	33.08 ± 2.91	28.82 ± 3.20	56.38 ± 1.02	53.27 ± 2.51
BTGF	53.39 ± 8.71	51.58 ± 14.20	71.88 ± 17.74	70.13 ± 20.45	48.67 ± 12.11	49.66 ± 16.04	70.35 ± 13.93	68.72 ± 15.11	11.70 ± 4.24	6.09 ± 1.84	45.93 ± 1.12	34.99 ± 2.07
BMGC	73.27 ± 1.71	77.68 ± 1.76	91.90 ± 0.72	91.90 ± 0.72	69.24 ± 0.19	75.54 ± 0.28	89.60 ± 0.15	88.92 ± 0.17	71.46 ± 0.42	73.93 ± 0.48	91.53 ± 0.17	92.44 ± 0.15
NGCE	76.56 ± 0.46	81.02 ± 0.52	93.22 ± 0.20	93.22 ± 0.21	70.60 ± 0.80	74.52 ± 0.98	89.27 ± 0.46	88.85 ± 0.47	53.08 ± 0.98	42.81 ± 1.36	78.18 ± 0.73	80.01 ± 0.74
InfoMGF	71.79 ± 4.51	75.01 ± 6.89	90.74 ± 3.06	90.83 ± 3.06	68.00 ± 1.24	70.58 ± 2.44	87.45 ± 1.13	87.07 ± 1.10	72.93 ± 0.68	75.46 ± 1.11	92.10 ± 0.39	93.01 ± 0.34
CGAF	**76.83 ± 0.47**	**82.06 ± 0.60**	**93.65 ± 0.24**	**93.68 ± 0.24**	**71.56 ± 1.15**	**75.99 ± 1.49**	**89.95 ± 0.68**	**89.54 ± 0.70**	**73.52 ± 0.39**	**76.68 ± 0.48**	**92.53 ± 0.17**	**93.37 ± 0.17**

Table 7. Mean differences from the strongest evaluated baseline in each column of Table 6, with descriptive 95% confidence intervals (CI), in percentage points. Welch intervals are used on ACM/DBLP and paired intervals on Yelp, where runs share pretrained backbones. Each method has five runs. Intervals concern fixed-graph optimization variability and are unadjusted for multiple comparisons and baseline selection.

Dataset	**Metric**	**Comparator**	**Difference**	**95% CI**
ACM	NMI	NGCE	0.27	[−0.42, 0.95]
ACM	ARI	NGCE	1.04	[0.22, 1.86]
ACM	ACC	NGCE	0.43	[0.11, 0.75]
ACM	F1	NGCE	0.46	[0.14, 0.79]
DBLP	NMI	NGCE	0.96	[−0.52, 2.43]
DBLP	ARI	BMGC	0.45	[−1.39, 2.28]
DBLP	ACC	BMGC	0.35	[−0.48, 1.19]
DBLP	F1	BMGC	0.61	[−0.24, 1.46]
Yelp	NMI	InfoMGF-RA	0.58	[−0.41, 1.57]
Yelp	ARI	InfoMGF-RA	1.22	[−0.19, 2.63]
Yelp	ACC	InfoMGF-RA	0.43	[−0.07, 0.92]
Yelp	F1	InfoMGF-RA	0.36	[−0.07, 0.80]

The ACM differences in ARI, ACC, and macro-F1 have positive lower confidence bounds at the nominal 95% level. The other nine intervals include zero, including the three NMI intervals: $[-0.42, 0.95]$ on ACM, $[-0.52, 2.43]$ on DBLP, and $[-0.41, 1.57]$ on Yelp, in percentage points. Under the pointwise interpretation defined in Section 4.4, the results show consistently positive observed mean differences with varying estimation precision.

The strongest baseline changes across datasets, emphasizing the need to accommodate different relation structures. NGCE leads the comparison methods on ACM, whereas BMGC leads three metrics on DBLP and InfoMGF leads all four on Yelp. CGAF exceeds the strongest NMI baseline on each dataset—NGCE on ACM and DBLP, and InfoMGF on Yelp—by 0.27, 0.96, and 0.58 percentage points, respectively. On ACM, the larger improvement in ARI (1.04 points) accompanies the NMI gain, showing improved pairwise community agreement even where the baseline NMI is already high. On Yelp, the gains

in both ARI (1.22 points) and macro-F1 (0.36 points) extend the improvement to pairwise structure and category-level prediction quality.

Compared with InfoMGF, CGAF increases mean NMI by 5.04, 3.55, and 0.58 points on ACM, DBLP, and Yelp. Its NMI standard deviation is lower on all three datasets, decreasing from 4.51 to 0.47 on ACM and from 0.68 to 0.39 on Yelp. These differences reflect the combined effects of prototype assignments, community objectives, attentive fusion, and joint training. The following ablations examine the individual contributions.

5.2. Ablation Study

Six ablation variants examine the components that connect community evidence, relation fusion, and partition learning. The first four modify the fusion mechanism, and the last two remove individual community objectives. Table 8 specifies the changes, with the remaining architecture and coefficients kept as in CGAF.

Table 8. Ablation variants and their corresponding design questions.

Variant	Modification and Purpose
w/o community evidence	Set $\Delta a_{iv} = 0$. Tests community evidence beyond ordinary node-specific attention.
w/o residual fusion ($\gamma = 0$)	Set $Z^* = Z_f$. Tests the relation-fusion extension while retaining the trainable backbone and prototype head.
w/o degree correction	Remove b^v from Equation (8). Tests background correction while retaining neighborhood averaging.
Global layer attention	Average $a_{iv} + \Delta a_{iv}$ over nodes before softmax. Tests graph-wide weights against node-specific weights.
w/o self-training	Set $\lambda_{\mathrm{ST}} = 0$. Tests the sharpened-target KL objective with the prototype readout retained.
w/o modularity	Set $\lambda_{\mathrm{M}} = 0$. Tests direct structural supervision of final memberships.

The variants distinguish evidence construction from structural supervision. Removing degree correction changes the input to the attention network, whereas removing modularity changes the objective applied to final memberships. Likewise, setting $\gamma = 0$ removes the entire relation-fusion adjustment, while removing community evidence preserves representation-based attention.

The full CGAF model obtains the highest reported NMI and ARI means among the six ablation variants on all three datasets. Table 9 reports the results for the controls defined in Table 8.

Table 9. Ablation comparison on ACM, DBLP, and Yelp (mean ± sample standard deviation over five repetitions, in percent). Bold and underlined entries indicate the highest and second-highest means among the displayed variants, respectively.

	ACM		DBLP		Yelp	
Method	**NMI**	**ARI**	**NMI**	**ARI**	**NMI**	**ARI**
CGAF	**76.83 ± 0.47**	**82.06 ± 0.60**	**71.56 ± 1.15**	**75.99 ± 1.49**	**73.52 ± 0.39**	**76.68 ± 0.48**
w/o community evidence	75.74 ± 0.59	80.97 ± 0.66	70.59 ± 1.07	75.03 ± 1.39	72.51 ± 0.42	75.71 ± 0.52
w/o residual fusion ($\gamma = 0$)	<u>76.11 ± 0.23</u>	<u>81.12 ± 0.32</u>	70.66 ± 0.79	75.21 ± 1.08	72.33 ± 0.31	75.60 ± 0.40
w/o degree correction	75.84 ± 0.46	81.08 ± 0.55	70.51 ± 1.00	74.94 ± 1.41	72.51 ± 0.45	75.68 ± 0.53
Global layer attention	75.90 ± 0.63	80.99 ± 0.69	70.57 ± 0.97	75.06 ± 1.32	71.96 ± 0.30	75.06 ± 0.35
w/o self-training	75.74 ± 0.51	81.05 ± 0.66	<u>70.92 ± 0.99</u>	<u>75.61 ± 1.24</u>	61.14 ± 12.48	62.58 ± 14.41
w/o modularity	75.73 ± 0.91	80.96 ± 1.00	69.98 ± 1.22	74.14 ± 1.68	<u>72.61 ± 0.47</u>	<u>75.89 ± 0.63</u>

Community evidence improves fusion beyond attention based only on representation compatibility. Removing the evidence correction reduces mean NMI by 1.09, 0.97, and 1.01 points on ACM, DBLP, and Yelp, respectively. Removing degree correction also lowers both reported metrics across the three datasets. The consistent reductions support using degree-corrected neighborhood memberships to guide relation weighting before community assignment.

Node-specific weighting and residual fusion address different aspects of relation integration. The global-attention control gives every node the same relation weights and shows its largest NMI reduction on Yelp (1.56 points), where three relations offer distinct neighborhood contexts. Setting $\gamma = 0$ instead removes the relation-specific adjustment entirely and reduces mean NMI by 0.72–1.19 points. These comparisons support adding node-specific relation fusion to the fused-graph representation.

Self-training and modularity provide complementary signals for learning the final partition. Without self-training, Yelp exhibits a marked reduction in mean NMI and a larger standard deviation, highlighting the contribution of sharpened membership targets to consistent optimization on this dataset. The reduction on ACM and DBLP is smaller, showing that the influence of this objective depends on the graph. Removing modularity produces its largest NMI decrease on DBLP (1.58 points), consistent with the role of an explicit structural objective alongside representation learning. Retaining both objectives provides the strongest reported means across the three datasets.

5.3. Parameter Sensitivity Analysis

Parameter analysis examines how the community objectives and residual fusion affect the learned partition. Each sweep varies one parameter over the values in Table 10 and fixes the others at the main configuration. We use five additional seeds, 31–35, on each dataset. Settings within a dataset–seed pair share one pretrained backbone and one prototype initialization, followed by 100 joint-training epochs with a freshly initialized optimizer. The four sweeps reuse the same default run, yielding 25 unique settings per pair and 375 continuation runs in total. Figure 3 reports NMI means and sample standard deviations for the four sweeps. Changes below are measured relative to the default configuration from this parameter experiment.

Table 10. Parameter grids and the mechanism examined by each sweep.

Parameter	Candidate Values	Mechanism
λ_{ST}	$0, 0.05, 0.1, 0.25, 0.5, 1, 2$	Strength of membership self-training
λ_{M}	$0, 0.01, 0.03, 0.05, 0.1, 0.3, 0.5$	Structural supervision of communities
λ_{CL}	$0, 0.1, 0.25, 0.5, 1, 2, 5$	Contrastive learning during joint training
γ_0	$0.01, 0.05, 0.1, 0.2, 0.4, 0.7, 0.9$	Initialization of residual fusion

5.3.1. Weight of Self-Training

Self-training improves the quality and consistency of community assignments on Yelp. Increasing λ_{ST} from zero to its default value of 0.5 raises mean NMI by 7.05 points and mean ARI by 7.74 points, while the NMI standard deviation falls from 10.33 to 0.68. The mean assignment entropy also decreases from 0.0982 to 0.0699, with all three communities occupied in every run. Thus, the sharpening objective improves reference-partition agreement while reducing variation across initializations. Across the five tested weights $\{0.1, 0.25, 0.5, 1, 2\}$, the NMI and ARI means on all three datasets remain within one percentage point of the default values.

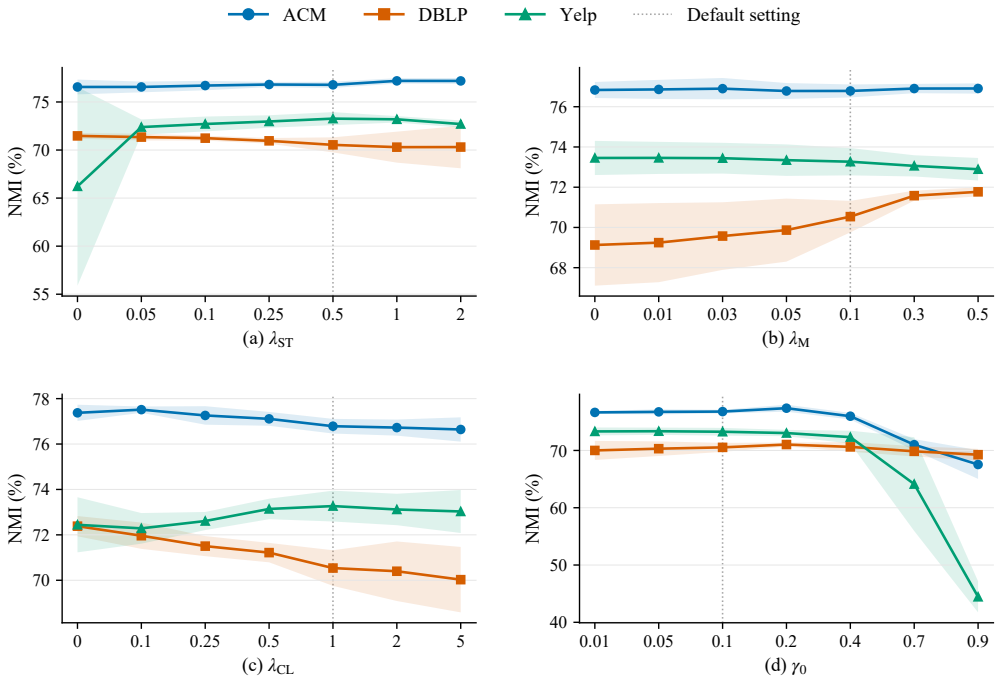


Figure 3. NMI sensitivity to (**a**) self-training weight, (**b**) modularity weight, (**c**) joint contrastive weight, and (**d**) residual-coefficient initialization. Lines and shaded bands denote means and sample standard deviations over five seeds. One parameter is varied at a time, with shared pretraining and prototype initialization within each dataset–seed pair. Dotted lines mark the default settings, whose runs are reused across panels. Ticks enumerate the tested values with categorical spacing. Contrastive pretraining is retained at $\lambda_{CL} = 0$, and γ remains trainable throughout the initialization sweep.

5.3.2. Weight of Modularity

The modularity objective provides useful structural supervision on DBLP. Raising λ_M from zero to 0.1 improves mean NMI and ARI by 1.41 and 2.35 points, respectively. The average modularity of the final hard partition increases from 0.5437 to 0.5551, showing that better agreement with the reference communities accompanies stronger structural concentration. At $\lambda_M = 0.5$, DBLP reaches an NMI of 71.77 and an ARI of 76.86. In contrast, the two metrics on ACM and Yelp remain within one point of their default means across all seven tested weights. The structural objective therefore has a dataset-dependent influence, complementing the representation and self-training objectives most clearly on DBLP.

5.3.3. Weight of Joint Contrastive Learning

The contrastive weight controls the balance between preserving graph representations and adapting them to community objectives. Pretraining is identical for every setting, including $\lambda_{CL} = 0$, which removes the contrastive objective only during joint training. ACM and Yelp retain NMI and ARI means within one point of their defaults over all seven tested weights. DBLP benefits from a smaller joint contrastive weight, reaching an NMI of 72.38 and an ARI of 77.90 at zero. The DBLP results favor stronger adaptation to the community objectives during joint training, whereas ACM and Yelp change little over the tested weights.

5.3.4. Initialization of Residual Fusion

Small residual initializations preserve the fused representation while allowing node-specific relation information to refine the partition. At the four tested values $\gamma_0 \in \{0.01, 0.05, 0.1, 0.2\}$, the NMI and ARI means on all three datasets remain within one percentage point of the default configuration. Larger values place more weight on relation embeddings from the start and reduce partition quality particularly on ACM and Yelp. The

results favor an initialization dominated by the fused-graph embedding, with a smaller contribution from attentive relation fusion.

The learned residual coefficient adjusts locally around its initialization during the 100-epoch joint stage. Starting from $\gamma_0 = 0.1$, its final means are 0.1091, 0.1092, and 0.1024 on ACM, DBLP, and Yelp, respectively. The default initialization lies within the stable set of tested small values and retains the intended emphasis on the fused-graph representation. All sweeps use the original common default configuration reported in Section 4.

5.3.5. Guidance for Unlabeled Deployment

The common configuration $(\lambda_{\mathrm{CL}}, \lambda_{\mathrm{ST}}, \lambda_{\mathrm{M}}, \lambda_{\mathrm{bal}}) = (1, 0.5, 0.1, 0.01)$ and $\gamma_0 = 0.1$ provides a reproducible starting point for unlabeled deployment. For self-training, the five tested positive weights from 0.1 to 2 stay within one percentage point of default NMI and ARI on all three graphs, supporting 0.5 as a common default. The four tested residual initializations $0.01, 0.05, 0.1, 0.2$ show the same descriptive stability. These observations favor a small initial contribution from relation fusion while the community pathway adapts during joint training.

Modularity and joint contrastive weights have dataset-dependent effects, with DBLP favoring stronger adaptation to community objectives. For unlabeled graphs, we recommend starting with the common defaults and monitoring loss trajectories, community occupancy, and assignment entropy for collapse or unstable optimization. These diagnostics assess optimization behavior; assessing semantic agreement requires external reference information. Likewise, higher modularity measures structural concentration, whose relationship to semantic communities depends on the application. The reported stability covers the tested discrete values varied one at a time, with λ_{bal} fixed. This protocol provides concrete initial settings and diagnostic quantities for applying CGAF to a new graph.

5.4. Execution Time and Memory Versus Network Size

To isolate the resource effect of node count, we profile CGAF at five sizes: $N = 1000$, 2000, 4000, 6000, and 8000, using controlled three-layer inputs. Each layer is an independently permuted undirected ring lattice of degree 20, so it has $10N$ edges. Shared features are row-normalized Gaussian vectors with $d_x = 128$. We fix $h = 128$, $d = d_p = 64$, $K = 4$, $k_g = 15$, two propagation steps, the default loss coefficients and dropout, and learning rate 0.001 in both stages. The generated inputs serve as computational workloads with fixed architectural dimensions and input degree across sizes.

Each size runs in a fresh process on the RTX 4060 Laptop GPU and software environment described in Section 4, under Windows with the WDDM driver model. For each training stage, two warmup updates precede five measured updates; the joint stage follows prototype initialization from the profiling pretraining output. Each step includes forward computation, loss evaluation, backpropagation, and the Adam update, with CUDA synchronization at both timing boundaries. Data construction, preprocessing, and prototype initialization are outside step timing. Peak CUDA tensor allocation covers setup, initialization, and both stages; process host working-set memory is sampled every 10 ms. The profile uses dense float32 tensors without checkpointing, explicit CPU tensor offloading, or mixed precision. These measurements characterize per-update cost and peak allocation under the specified short profiling schedule.

Table 11 shows how computation and memory grow at fixed input degree. From 1000 to 4000 nodes, peak CUDA allocation increases from 356 to 3993 MiB, and the median joint update increases from 0.044 to 0.350 s. At 6000 and 8000 nodes, allocator-reported demand reaches 8653 and 15121 MiB, exceeding the approximately 8188 MiB dedicated GPU capacity; median joint steps take 8.111 and 36.294 s. Windows system-memory fallback

can support allocations above dedicated capacity at reduced speed. NVIDIA documents this behavior at https://nvidia.custhelp.com/app/answers/detail/a_id/5490 (accessed on 14 September 2026). The CUDA column records tensor allocation demand, while host RSS records the process working set; physical residency and driver-managed shared memory are outside these measurements, so the two columns are reported separately. The sharp increase beyond 4000 nodes identifies a practical capacity constraint for the dense implementation on this hardware. Together with the quadratic terms in Section 3.6, the profile supplies quantitative guidance for resource planning and identifies graph construction and full-node contrasts as targets for scalable implementation.

Table 11. Controlled dense CGAF resource profile. Step times are medians [minimum, maximum] over five synchronized updates after two warmup updates per stage, in seconds. Memory is reported in MiB: peak PyTorch CUDA tensor allocation and sampled process host RSS over setup and both stages. CUDA allocation is not dedicated physical-VRAM occupancy; values marked † exceed the card's approximately 8188 MiB dedicated capacity on this Windows/WDDM system.

Nodes	Pretraining Step (s)	Joint Step (s)	CUDA Allocation	Host RSS
1000	0.029 [0.028, 0.042]	0.044 [0.035, 0.051]	356	1300
2000	0.074 [0.073, 0.076]	0.085 [0.080, 0.087]	1112	1339
4000	0.334 [0.330, 0.335]	0.350 [0.346, 0.351]	3993	1388
6000	6.319 [6.017, 6.802]	8.111 [7.905, 8.540]	8653 †	2688
8000	37.849 [37.818, 37.882]	36.294 [36.151, 45.301]	15,121 †	9592

5.5. Practical Scope and Limitations

Prescribed community count.

CGAF learns a partition conditional on a supplied count K. We use the benchmark class counts: 3 for ACM, 4 for DBLP, and 3 for Yelp. These experiments therefore assess community assignment at a known granularity. A count below the number of relevant groups can merge communities; a larger count can split groups or leave prototypes sparsely occupied. The occupancy regularizer redistributes soft membership mass among the prescribed prototypes and does not estimate their number.

When K is unknown, it must be selected before fitting the current model. Domain knowledge can define a useful resolution, while spectral estimates from a combined graph or model-selection criteria can provide candidate counts [1]. Partition stability across initializations and structural quality can then inform the choice. These options would require a separate model-selection procedure. Incorporating a community-count estimator would extend the framework to applications without a prescribed partition granularity.

Sparse, noisy, and missing relations.

For unit-normalized representations, let $r_i = (1-\gamma)z_i^f + \gamma \sum_v \beta_{iv} z_i^v$ denote the vector before the final normalization. Convexity gives $\|r_i - z_i^f\|_2 \leq 2\gamma$. If only relation v's embedding changes by δz_i^v, holding z_i^f, β, and γ fixed, then $\|\delta r_i\|_2 \leq \gamma \beta_{iv} \|\delta z_i^v\|_2$. These bounds characterize the residual path under fixed-context perturbations. End-to-end sensitivity also involves changes to the learned fused graph, source memberships, and attention, together with possible amplification by final row normalization when $\|r_i\|_2$ is small. The global coefficient γ controls the overall residual contribution, while β allocates relation weights per node. This division gives the mechanism both a shared fusion scale and node-specific relation weighting.

An isolated node receives zero evidence from that relation, and an empty relation receives zero evidence throughout. Centering then sets the evidence correction to zero,

while base attention and attribute-driven graph learning continue to supply representations. Thus, an empty raw layer still participates through its feature-based pathway. Sparse neighborhoods provide fewer observations, so their evidence retains sampling uncertainty after background correction. The implementation uses a fixed set of aligned relation inputs: empty topology preserves an input slot, whereas removing the slot changes the architecture. Training-time edge dropout supplies augmented graph observations. Quantitative evaluation across corruption levels and missing-layer patterns is a direction for future work, building on the residual-path analysis and the explicit treatment of empty relations described here.

6. Conclusions

This paper presented CGAF for community detection in attributed multiplex networks. The method uses degree-corrected neighborhood memberships to adjust node-specific relation attention. A learnable residual coefficient combines relation embeddings with the fused-graph representation, and shared prototypes connect this fusion to community assignment. Joint optimization allows the community objectives to update both the representations and the weights used to integrate them.

On ACM, DBLP, and Yelp, CGAF achieved the highest observed means across the four reported metrics among eight comparison methods under the stated protocols. Confidence intervals quantify uncertainty in the mean differences, while ablation and parameter analyses show the contributions of community evidence, structural supervision, and residual fusion. The common loss coefficients and tested small residual initializations provide practical starting settings. Resource profiling and the analysis of the model's assumptions motivate future work on sparse computation, automatic community-count selection, and partially observed relations. CGAF provides an explicit, differentiable way to use neighborhood community information when combining relations for multiplex clustering.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L.; formal analysis, Y.L.; investigation, Y.L.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and H.C.; visualization, Y.L.; supervision, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: Publication of this article is supported by the Top Talent Training Project of Henan Province (Grant No. 244500510012).

Institutional Review Board Statement: Not applicable. This study did not involve humans or animals.

Informed Consent Statement: Not applicable.

Data Availability Statement: The processed ACM, DBLP, and Yelp datasets are publicly available from the InfoMGF repository at https://github.com/zxlearningdeep/InfoMGF (accessed on 14 September 2026). The reported experimental results are included in this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fortunato, S.; Hric, D. Community detection in networks: A user guide. *Phys. Rep.* **2016**, *659*, 1–44. [CrossRef]
2. Liu, Y.; Peng, Y.; Chen, H.; Zhang, J.; Li, S. BCDAN: A balanced method for community detection in attributed networks. *Neurocomputing* **2025**, *644*, 130411. [CrossRef]
3. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
4. Tsitsulin, A.; Palowitch, J.; Perozzi, B.; Müller, E. Graph Clustering with Graph Neural Networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–21.

5. Mucha, P.J.; Richardson, T.; Macon, K.; Porter, M.A.; Onnela, J.P. Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. *Science* **2010**, *328*, 876–878. [CrossRef] [PubMed]
6. Jing, B.; Park, C.; Tong, H. HDMI: High-order Deep Multiplex Infomax. In *Proceedings of the Web Conference 2021*; ACM: New York, NY, USA, 2021; pp. 2414–2424. [CrossRef]
7. Wang, B.; Cai, X.; Xu, M.; Xiang, W. A graph-enhanced attention model for community detection in multiplex networks. *Expert Syst. Appl.* **2023**, *230*, 120552. [CrossRef]
8. Shen, Z.; Wang, S.; Kang, Z. Beyond Redundancy: Information-aware Unsupervised Multiplex Graph Structure Learning. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2024; Volume 37, pp. 31629–31658. [CrossRef]
9. Newman, M.E.J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef]
10. Pan, E.; Kang, Z. Multi-view Contrastive Graph Clustering. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 2148–2159.
11. Qian, X.; Li, B.; Kang, Z. Upper Bounding Barlow Twins: A Novel Filter for Multi-Relational Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2024; Volume 38, pp. 14660–14668. [CrossRef]
12. Shen, Z.; He, H.; Kang, Z. Balanced Multi-Relational Graph Clustering. In *Proceedings of the 32nd ACM International Conference on Multimedia*; ACM: New York, NY, USA, 2024; pp. 4120–4128. [CrossRef]
13. Ren, Y.; Ke, J.; Wen, Z.; Wu, T.; Yang, Y.; Pu, X.; He, L. Multi-View Graph Clustering via Node-Guided Contrastive Encoding. In *Proceedings of the 42nd International Conference on Machine Learning*; PMLR: London, UK, 2025; Volume 267, pp. 51421–51435.
14. Cheng, J.; He, C.; Fei, X.; Hao, T.; Ma, H.; Tang, Y. Variational graph filter autoencoder for uncovering community structure in multiplex networks. *Pattern Recognit.* **2026**, *172*, 112668. [CrossRef]
15. Veličković, P.; Fedus, W.; Hamilton, W.L.; Liò, P.; Bengio, Y.; Hjelm, R.D. Deep Graph Infomax. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
16. Park, C.; Kim, D.; Han, J.; Yu, H. Unsupervised Attributed Multiplex Network Embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2020; Volume 34, pp. 5371–5378. [CrossRef]
17. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised Deep Embedding for Clustering Analysis. In *Proceedings of the 33rd International Conference on Machine Learning*; PMLR: London, UK, 2016; Volume 48, pp. 478–487.
18. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, Macao, China, 10–16 August 2019; pp. 3670–3676. [CrossRef] [PubMed]
19. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*; PMLR: London, UK, 2019; Volume 97, pp. 6861–6871.
20. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.