Supplementary Material

A Algorithm Implementation

To have a clear understanding of the contribution of the proposed model, we present a comparison with existing methods that have different core components in Table 1. We also summarize the detailed training process for our CCA-AGC model in Algorithm 1. It is worth mentioning that during joint training, the target distribution $\mathbf{P}$ serves as the target label, but also depends on the predicted soft assignment $\mathbf{Q}$. To avoid instability, we do not update $\mathbf{P}$ at each iteration in practice, but $\mathbf{P}$ is updated every T iterations according to Eq. (5).

B Additional Experimental Details

In this section, we first describe some details about the datasets and parameter settings, and then give some additional experimental results as a complement.

B.1 Datasets

All datasets we used to verify the clustering performance in the experiments are available in the PyTorch Geometric libraries[1]. We provided the basic statistics of these datasets in Table 2.

- **Cora, CiteSeer, and PubMed**. They are representative citation networks, where nodes and edges denote documents and citations. The node features are bag-of-words for documents, and labels indicate the field of documents.
- **WikiCS**. It is a reference network constructed based on Wikipedia. The nodes correspond to articles about computer science, and the edges are hyperlinks between them. Nodes are labeled with ten branches of the field, and node features are calculated as the average of pre-trained GloVe word embeddings in each article.
- **Amazon-Computers and Amazon-Photo**. They represent co-purchase relationships constructed based on Amazon. Nodes are goods, and edges indicate that two goods are frequently bought together. Each node has a sparse bag-of-words encoding and is labeled with a product category.
- **Coauthor-CS**. It is a co-authorship graph based on the Microsoft Academic Graph from the KDD Cup 2016 Challenge, where nodes represent authors. Two nodes are connected if they co-author a paper. Node features represent paper keywords for each author's papers, and the label indicates the author's primary research field.

[1] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html

Table 1: Technical components comparison with existing methods.

Model	Framework	Contrasting views	Augmentation strategies	Clustering
GRACE	Contrastive	Two augmented graphs	Random perturbing	Two-stage
GCA	Contrastive	Two augmented graphs	Adaptive perturbing	Two-stage
MVGRL	Contrastive	Two augmented sub-sample graphs	Diffusion graph	Two-stage
CCGC	Contrastive	-	Unshare encoders	Two-stage
SDCN	Autoencoder	-	-	One-stage
DAGC	Autoencoder	-	-	One-stage
CCA-AGC	Contrastive	Original graph + generated graph	kNN graph + EBC	One-stage

Table 2: Statistics of real-word attributed graphs.

Dataset	#Nodes	#Edges	#Attributes	#Clusters
Cora	2708	5429	1433	7
CiteSeer	3327	4732	3703	6
PubMed	19717	44338	500	3
WikiCS	11701	216123	300	10
AmazonCom	13752	245861	767	10
Amazon-Photo	7650	119081	745	8
Coauthor-CS	18333	81894	6805	15

Algorithm 1 Training algorithm of CCA-AGC

Input: An attributed graph $G = (\mathbf{A}, \mathbf{X})$, number of clusters K, maximum number of iterations $MaxIter$, update interval T.
Output: Clustering result C.

Generate a kNN graph and filter its edges with EBC;
Construct a contrasting view G' by masking node features of the updated kNN graph;
Pre-train multilevel contrast by minimizing Eq. (4) to get initial parameters and representations $\mathbf{H}$ and $\mathbf{H}'$; {Pre-training}
Run K-means on $\mathbf{H}$ and $\mathbf{H}'$ to get initial cluster centers μ and μ' of two contrasting views; {Pre-training}
for $iter \in \{0, 1, \cdots, MaxIter\}$ **do** {Training}
    Learn encoding representations $\mathbf{H}, \mathbf{H}'$ and projecting representations $\mathbf{Z}, \mathbf{Z}'$ of nodes;
    Compute soft assignments $\mathbf{Q}$ and $\mathbf{Q}'$ of two views;
    if $iter\%T == 0$ **then**
        Update target distribution $\mathbf{P}$;
    end if
    Calculate the multilevel contrast loss $\mathcal{L}_{tra}$ via Eq. (4);
    Calculate the clustering loss $\mathcal{L}_{clu}$ via Eq. (7);
    Update the whole model by minimizing Eq. (8);
end for
return C, where $c_i = \arg\max_k q_{ik} (i = 1, \cdots, n; k = 1, \cdots, K)$.

B.2 Implementation Details

We perform a grid search to select hyper-parameters for each dataset in the following search space: the dimension of encoding and projecting representations

Table 3: Hyperparameter settings of the proposed CCA-AGC model.

Dataset	Encoding dimension	Projecting dimension	Activation function	Learning rate	kNN	p_e	p_m	Epoch	T
Cora	512-256	1024-1024	ReLu	0.0001	0	0.85	0.1	200	1
CiteSeer	1024-512	1024-1024	PReLu	0.0005	1	0.65	0.4	300	1
PubMed	1024-512	512-512	ReLu	0.001	5	0.9	0.2	200	1
WikiCS	1024-1024	128-128	PReLu	0.005	0	0.01	0.2	200	20
AmazonCom	128-128	1024-1024	PReLu	0.0005	10	0.65	0	200	200
Amazon-Photo	512-128	1024-1024	ReLu	0.00003	6	0.85	0	200	20
Coauthor-CS	256-256	1024-1024	PReLu	0.001	0	0.5	0	200	200

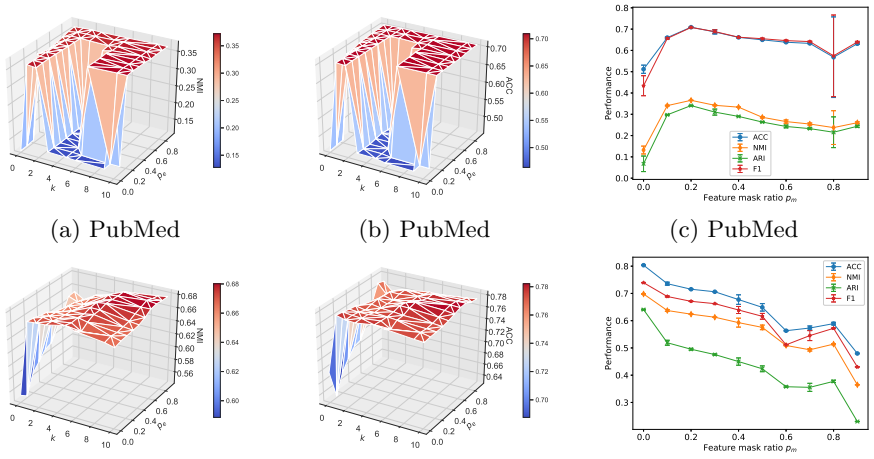
is searched in $\{128, 256, 512, 1024\}$; the learning rate is searched in $\{1e-5, 3e-5, 5e-5, 1e-4, 5e-4, 0.001, 0.005, 0.01\}$; the probabilities of edge filtering p_e and feature masking p_m are tuned from 0.1 to 0.9; the k of our kNN graph is tuned in $\{1, 2, 3, 5, 6, 9, 10\}$; and the interval T for updating target distribution $\mathbf{P}$ is tuned in $\{1, 20, 50, 100, 200\}$. All experiments are conducted with PyTorch 1.10.0 and PyTorch Geometric 2.0.2, and all dataset-specific configurations are provided in Table 3 for reproducibility.

B.3 Sensitivity Analysis

Analysis of augmentation ratio We study the influence of some critical hyperparameters on clustering performance, namely k, p_e, and p_m, which determine the generation of the kNN graph and the contrasting view in the CCA-AGC model. We first set the mask ratio of node attributes p_m as a constant for brevity in the visualization and show the influence of the other two parameters. Due to space limitations, we only take PubMed and Amazon-Photo datasets as examples to present the results in ACC and NMI metrics in Figs. 1(a)-(b), (d)-(e), and other cases have similar trends. Here, $k = 0$ means removing edges from the original graph rather than the generated kNN graph. Then we fix k and p_e and further show the influence of the feature mask ratio p_m in Figs. 1(c), (f).

It can be observed that the performance is relatively stable when k and p_e are large. A larger k helps identify more node pairs with similar features, leading to a denser graph. The larger p_e filters out more edges that are more likely to connect different clusters, resulting in a clearer cluster structure. However, when we remove edges from the original graphs (i.e., $k = 0$), there is a high probability that the inherent graph structures will be destroyed, making it challenging to obtain satisfactory clusters. Additionally, Figs. 1(c), (f) show that a large p_m leads to poor clustering results. We think this is because the topology of the generated contrasting view mainly considers feature similarity, so preserving the original feature information as much as possible, i.e., using a smaller p_m, is more beneficial for the proposed model.

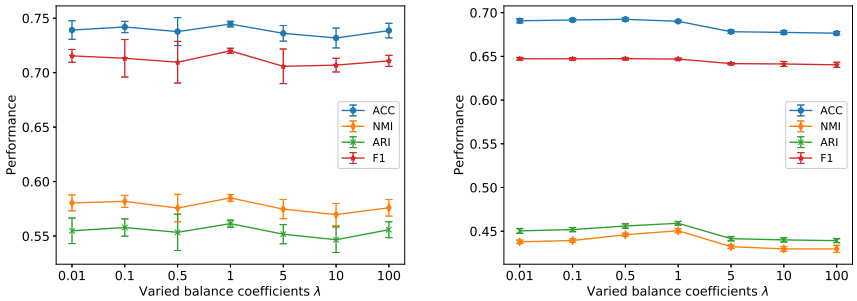
Analysis of loss coefficient We further investigate the impact of the coefficient λ on clustering performance, which balances the weight of the contrastive loss for



(a) PubMed (b) PubMed (c) PubMed

(d) Amazon-Photo (e) Amazon-Photo (f) Amazon-Photo

Fig. 1: Clustering performance of CCA-AGC model under varied augmentation hyperparameters.



(a) Cora (b) CiteSeer

Fig. 2: Clustering performance of CCA-AGC model under different balance coefficients λ.

encoding and projecting representations during joint training. Fig. 2 shows the clustering results with varying coefficients λ on the Cora and CiteSeer datasets.

According to Fig. 2, we see that the balance coefficient λ has little impact on the overall clustering performance in most cases. It implies that our CCA-AGC model is not sensitive enough to the choice of λ. However, the clustering performance decreases to some extent once $\lambda > 1$. This may be because a larger λ could cause the other two parts of the loss to be ignored during joint training, thus negatively affecting overall results. Consequently, for simplicity, we set λ to 1 for all datasets in our experiments.