Appendix - Synergistic Dual Proxies: Enhancing Cohesion and Separability in Deep Graph Clustering

A EXPERIMENTAL SETUP DETAILS

In this section, we provide additional details on the experimental setup, including framework architectures, hyperparameter settings, and computing infrastructure.

A.1 Framework Architectures

In our DP-Net framework, the GNN-based node encoder f leverages a standard Graph Convolutional Network (GCN) model [3]. This encoder, uniformly employed across all datasets, learns node representations that capture the underlying graph structure and node features. These representations are subsequently refined by a linear projection layer to enhance their suitability for clustering.

GCN Layer. Each layer of the GCN encoder is defined as:

$$\mathbf{H}^{(l+1)} = \text{PReLU}\left(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix, $\mathbf{D}$ is the degree matrix, $\mathbf{H}^{(l)} \in \mathbb{R}^{n\times d}$ is the node feature matrix at layer l (with $\mathbf{H}^{(0)} = \mathbf{X}$ as the input feature matrix), and $\mathbf{W}^{(l)}$ is the learnable weight matrix at layer l. The activation function used is PReLU [2]. The GCN layers are implemented using the `GCNConv` module from the PyTorch Geometric (PyG) library [1] for efficient computation and optimized performance.

Projection Layer. Following the GCN encoder, we incorporate a linear projection layer to further refine the final node representations $\mathbf{H}^{(L)}$ from the L-th layer:

$$\mathbf{Z} = \mathbf{H}^{(L)}\mathbf{W}_{\text{proj}} + \mathbf{b}_{\text{proj}}, \tag{2}$$

where $\mathbf{Z} \in \mathbb{R}^{n\times d}$ is the projected representation, $\mathbf{W}_{\text{proj}}$ is the learnable weight matrix, and $\mathbf{b}_{\text{proj}}$ is the bias vector. This layer is implemented via `nn.Linear(d,d)` module in PyTorch.

A.2 Hyperparameter Settings

To ensure reproducibility, we detail the hyperparameter settings used in our experiments with the DP-Net framework, as summarized in Table 1. These settings fall into three categories: architecture settings, augmentation settings, and optimization settings. The architecture settings specify the structural design of the DP-Net framework: the number of GCN layers L, the hidden dimension d, the balancing hyperparameter α (which regulates the regularization term in Equation 9), and the Lagrange multiplier β (which optimizes clustering assignments in Equation 12). The augmentation settings define the parameters for data augmentation, namely the feature masking probabilities $p_{\text{feat},:}$ and edge removing probabilities $p_{\text{edge},:}$. The optimization settings control the training process, including the number of training epochs T, the batch size B in mini-batch training, the learning rate lr, and the weight decay wd.

A.3 Computing Infrastructure

To accommodate the diverse computational demands of our experiments, we employed two distinct hardware configurations tailored to dataset sizes. For smaller datasets, such as Cora, CiteSeer, Amazon Photo, Amazon Computers, and ogbn-arxiv, we utilized a Linux server (Ubuntu 20.04 LTS) equipped with 64 CPU cores (Intel Xeon Gold 6226R, 2.90GHz), 256GB of RAM, and an NVIDIA GeForce RTX 3090 GPU (24GB). For larger datasets, including Reddit, ogbn-products, and ogbn-papers100M, we used a Linux server (Ubuntu 22.04 LTS) featuring 64 CPU cores (Intel Xeon Gold 5218, 2.30GHz), 384GB of RAM, and four NVIDIA V100 GPUs (32GB). Both systems were supported by an identical software stack, comprising Python 3.8.19, PyTorch 1.8.1, PyG 2.1.0.post1, Numpy 1.23.4, and CUDA 11.1, ensuring a consistent experimental environment and facilitating reproducibility.

REFERENCES

[1] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*. 1026–1034.

[3] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Table 1: Hyperparameter configurations for experimental results in Table 2 and Table 3.

Dataset	Architecture settings				Augmentation settings				Optimization settings			
	L	d	α	β	$p_{\text{feat},1}$	$p_{\text{feat},2}$	$p_{\text{edge},1}$	$p_{\text{edge},2}$	T	B	lr	wd
Cora	1	512	0.1	10^2	0.3	0.3	0.3	0.3	500	—	3×10^{-4}	1×10^{-4}
CiteSeer	1	1536	0.1	10^3	0.1	0.1	0.5	0.5	200	—	4×10^{-4}	5×10^{-5}
Amazon Photo	1	512	25	1	0.0	0.0	0.0	0.4	500	—	9×10^{-5}	1×10^{-3}
Amazon Computers	1	1024	1.0	5	0.0	0.0	0.15	0.3	1500	—	5×10^{-5}	7×10^{-4}
Reddit	3	512	0.4	10^2	0.0	0.0	0.3	0.4	100	4096	4×10^{-4}	1×10^{-5}
ogbn-arxiv	3	512	0.8	10	0.0	0.0	0.5	0.6	200	—	1×10^{-4}	4×10^{-2}
ogbn-products	4	256	0.5	10^3	0.0	0.0	0.1	0.1	100	1024	2×10^{-4}	3×10^{-3}
ogbn-papers100M	3	256	0.4	10^2	0.0	0.0	0.1	0.1	100	1024	7×10^{-4}	1×10^{-2}