Feature-Centric Unsupervised Node Representation Learning Without Homophily Assumption (Supplementary Material)

Sunwoo Kim[1], Soo Yong Lee[1], Kyungho Kim[1], Hyunjin Hwang[1], Jaemin Yoo[2], Kijung Shin[1,2]
[1]Kim Jaechul Graduate School of AI, KAIST, Seoul, South Korea
[2]School of Electrical Engineering, KAIST, Daejeon, South Korea
{kswoo97, syleetolow, kkyungho, julia510, jaemin, kijungs}@kaist.ac.kr

A Proof of Theorem 1

In this section, we present a proof of Theorem 1 presented in the main paper.

Proof. The proof consists of the three steps: (1) deriving the functional form $\texttt{CS}(z; n, n_0, w)$, (2) deriving the functional form of $\texttt{LCS}(z; n, n_0, w)$, and (3) proving Theorem 1 based on the results of (1) and (2).

Step 1. Obtaining the functional form of $\texttt{CS}(z; n, n_0, w)$. For simplicity, we denote $\texttt{CS}(z; n, n_0, w)$ as $\texttt{CS}(z)$. Given that a node belongs to $\mathcal{C}_0$ or $\mathcal{C}_1$ uniformly at random, $P(v_i \in \mathcal{C}_0) = P(v_i \in \mathcal{C}_1) = 1/2$ hold. Therefore, $\texttt{CS}(z)$ can be rewritten as follows:

$$\underbrace{\frac{\mathbb{E}_x[\mathbb{I}[P(v_i \in \mathcal{C}_0|z_i) > P(v_i \in \mathcal{C}_1|z_i)]|v_i \in \mathcal{C}_0]}{2}}_{\text{For the case where } v_i \in \mathcal{C}_0} \quad (1)$$

$$+ \underbrace{\frac{\mathbb{E}_x[\mathbb{I}[P(v_i \in \mathcal{C}_1|z_i) > P(v_i \in \mathcal{C}_0|z_i)]|v_i \in \mathcal{C}_1]}{2}}_{\text{For the case where } v_i \in C_1}. \quad (2)$$

Throughout the proof, we rewrite $w := 1 - w$, since this formula enables the easier derivation of the result.

We derive the case of $v_i \in \mathcal{C}_0$ (Eq. (1)) and extends the result to $v_i \in \mathcal{C}_1$ (Eq. (2)). We first obtain the probability distribution of $z_i = \left(wx_i + \frac{1-w}{n}\left(\sum_{v_j \in N(v_i)} x_j\right)\right)$. Since the weighted summation of independent variables sampled from the Gaussian distributions also follows the Gaussian distribution, we can infer that z_i also follows the Gaussian distribution. Therefore, we only need to estimate the mean and variance of the corresponding distribution. For the mean of neighboring nodes' features, the following holds:

$$\frac{1}{n}\left(\sum_{v_j \in N(v_i)} x_j\right) \sim \mathcal{N}\left((2n_0 - n)\mu, \frac{\sigma^2}{n}\right). \quad (3)$$

Then, by aggregating the results of Eq. (3) and the distribution of x_i, we derive the probability distribution of z_i as follows:

$$\mathcal{N}\left(\frac{\mu\,(wn + (1-w)(2n_0 - n))}{n}, \sigma^2 w^2 + \frac{\sigma^2(1-w)^2)}{n}\right). \quad (4)$$

By using Eq. (4), we derive Eq. (1). To this end, we first rewrite $P(v_i \in \mathcal{C}_0|z_i)$ by using the Bayes rule as follows:

$$P(v_i \in \mathcal{C}_0|z_i) = \frac{P(z_i|v_i \in \mathcal{C}_0) \times P(v_i \in \mathcal{C}_0)}{P(z_i)}. \quad (5)$$

Therefore, $P(v_i \in \mathcal{C}_0|z_i) > P(v_i \in \mathcal{C}_1|z_i) \equiv P(z_i|v_i \in \mathcal{C}_0) > P(z_i|v_i \in \mathcal{C}_1)$ holds. Given that the distribution of z_i of $v_i \in \mathcal{C}_0$ and that of $v_i \in \mathcal{C}_1$ is symmetry for zero due to the symmetric characteristic of the Gaussian distribution, the following holds:

$$P(z_i|v_i \in \mathcal{C}_0) > P(z_i|v_i \in \mathcal{C}_1) \equiv z_i > 0. \quad (6)$$

Consequently, Eq. (1) is equivalent to $P(z_i > 0|v_i \in \mathcal{C}_0)$ due to the result of Eq. (6).

We now extend the result of Eq. (6) to the derivation of Eq. (2). Given the symmetry of the Gaussian distribution, the distribution of $z_i, v_i \in C_1$ is symmetric for zero with respect to that of $z_i, v_i \in C_0$. Therefore, z_i follows the following probabilistic distribution:

$$\mathcal{N}\left(-\frac{\mu\,(wn + (1-w)(2n_0 - n))}{n}, \sigma^2 w^2 + \frac{\sigma^2(1-w)^2)}{n}\right). \quad (7)$$

Moreover, for the same reason, we can derive that Eq. (2) is equivalent to $P(z_i < 0|v_i \in \mathcal{C}_1)$.

We now derive the exact functional form of $P(z_i > 0|v_i \in \mathcal{C}_0)$ and $P(z_i < 0|v_i \in \mathcal{C}_1)$. Given Eq. (4), Eq. (7), and the symmetry of the Gaussian distribution, the following holds:

$$P(z_i|v_i \in \mathcal{C}_0) = \Phi\left(\frac{\mu\,(wn + (1-w)(2n_0 - n))}{\sigma\sqrt{n^2w^2 + n(1-w)^2}}\right), \quad (8)$$

$$P(z_i|v_i \in \mathcal{C}_1) = 1 - \Phi\left(-\frac{\mu\,(wn + (1-w)(2n_0 - n))}{\sigma\sqrt{n^2w^2 + n(1-w)^2}}\right), \quad (9)$$

$$= \Phi\left(\frac{\mu\,(wn + (1-w)(2n_0 - n))}{\sigma\sqrt{n^2w^2 + n(1-w)^2}}\right), \quad (10)$$

where $\Phi(\cdot)$ is a C.D.F. of the standard Gaussian distribution. Thus, $\texttt{CS}(z) = \Phi\left(\frac{\mu(wn + (1-w)(2n_0 - n))}{\sigma\sqrt{n^2w^2 + n(1-w)^2}}\right)$ holds.

Step 2. Obtaining the functional form of $\texttt{LCS}(z; n, n_0, w)$. We now drive the functional form of latent-class separability. Similar to Step 1, we rewrite $\texttt{LCS}(z; n, n_0, w) := \texttt{LCS}(z)$ Note that the functional form $\texttt{LCS}(z)$ is defined as follows:

$$\frac{(\mathbb{E}_x[z_i|v_i \in \mathcal{C}_0] - \mathbb{E}_x[z_i|v_i \in \mathcal{C}_1])^2}{\frac{\mathbb{E}_x[(z_i - \mathbb{E}[z_i|v_i \in \mathcal{C}_0])^2|v_i \in \mathcal{C}_0] + \mathbb{E}_x[(z_i - \mathbb{E}[z_i|v_i \in \mathcal{C}_1])^2|v_i \in \mathcal{C}_1]}{2}}. \quad (11)$$

We again rewrite w as $w := 1 - w$ for the simplicity of this proof. By combining the results in Eq. (4) and (7), $\texttt{LCS}(z)$ is derived as follows:

$$\frac{(\mathbb{E}_x[z_i|v_i \in \mathcal{C}_0] - \mathbb{E}_x[z_i|v_i \in \mathcal{C}_1])^2}{\frac{Var(z_i|v_i \in \mathcal{C}_0) + Var(z_i|v_i \in \mathcal{C}_1)}{2}}, \quad (12)$$

$$= \frac{\frac{(2\mu(wn + (1-w)(2n_0 - n)))^2}{n^2}}{\sigma^2\left(w^2 + \frac{(1-w)^2}{n}\right)}, \quad (13)$$

$$\equiv \frac{(wn + (1-w)(2n_0 - n))^2}{n^2 w^2 + n(1-w)^2}. \quad (14)$$

Step 3. Deriving the inequalities. From the functional forms of $\texttt{CS}(z)$ (Eq. (8) and (10)) and $\texttt{LCS}(z)$ (Eq. (14)), we derive the inequalities stated in Theorem 1. Since the C.D.F. of the standard Gaussian distribution is a strictly increasing function, the following equivalence is sufficient to compare class separabilities of different w.

$$\texttt{CS}(z) \equiv \frac{(wn + (1-w)(2n_0 - n))}{\sqrt{n^2 w^2 + n(1-w)^2}}. \quad (15)$$

Here, note that Eq. (14) is the square of Eq. (15). Therefore, if Eq. (15) > 0 holds, then it is trivial that Theorem 1 holds. To achieve this, the following should hold:

$$w(2n - 2n_0) - (n - 2n_0) > 0, \quad (16)$$

$$\equiv w > \frac{n - 2n_0}{2n - 2n_0} \quad (17)$$

When $n = n_0$ holds, Eq. (16) > 0 holds, which demonstrates the theoretical results. In addition, Eq. (17) is the given condition of Theorem 1, which ensures that Eq. (15) > 0 holds in the setting of Theorem 1. Thus, we conclude the proof. □

B Dataset details

In this section, we provide details of leveraged datasets. Dataset statistics are provided in Table 1. Specifically, we report the (1) number of nodes, (2) number of edges, (3) number of features, (4) number of classes, (5) node homophily (Pei et al. 2020), and (6) edge homophily (Zhu et al. 2020). We leverage dataset implementation provided in Pytorch Geometric (Fey and Lenssen 2019).

B.1 Non-homophilic graph datasets

The *actor* dataset (Tang et al. 2009; Pei et al. 2020) is the actor-only induced subgraph of a film-director-actor-writer network from Tang et al. (2009). Each node represents an actor, and each edge connects two nodes whose corresponding actors co-appeared on the same Wikipedia page. The node features are keywords of the corresponding actor's Wikipedia page. The node label is a category of the corresponding actor, determined based on the words of the actor's Wikipedia page.

The *chameleon* dataset (Rozemberczki, Allen, and Sarkar 2021; Pei et al. 2020) is the Wikipedia page-page network on the chameleon topic. Each node represents a web page, and each edge connects two nodes whose corresponding pages are mutually linked in the web. The node features are informative nouns on the corresponding page. The node label is the categorized monthly traffic of the corresponding page.

The *squirrel* dataset (Rozemberczki, Allen, and Sarkar 2021; Pei et al. 2020) is the Wikipedia page-page network on the squirrel topic. Each node represents a web page, and each edge connects two nodes whose corresponding pages are mutually linked in the web. The node features are informative nouns on the corresponding page. The node label is the categorized-monthly traffic of the corresponding page.

The *cornell* dataset (Pei et al. 2020) is the website hyperlink network. Each node represents a web page, and each edge connects two nodes whose corresponding pages are hyperlinked in the web. The node features are bag-of-word features derived from the corresponding web page. The node label is the category of the corresponding page, one among student, project, course, staff, and faculty.

The *wisconsin* dataset (Pei et al. 2020) is the website hyperlink network. Each node represents a web page, and each edge connects two nodes whose corresponding pages are hyperlinked in the web. The node features are bag-of-word features derived from the corresponding web page. The node label is the category of the corresponding page, one among student, project, course, staff, and faculty.

The *texas* dataset (Pei et al. 2020) is the website hyperlink network. Each node represents a web page, and each edge connects two nodes whose corresponding pages are hyperlinked in the web. The node features are bag-of-word features derived from the corresponding web page. The node label is the category of the corresponding page, one among student, project, course, staff, and faculty.

The *Penn94* dataset (Lim et al. 2021) is the social network from Facebook. Each node represents a user, and each edge connects two nodes whose corresponding users have friendships on Facebook. The node features are user profiles including education and residential information. The node label is the gender of the corresponding user.

The *Flickr* dataset (Zeng et al. 2020) is the image relation dataset. Each node represents an image, and each edge connects two nodes whose corresponding images share common characteristics (e.g., geometric location). The node features are bag-of-word features derived from the corresponding image's text description. The node label is the tag-related category of the corresponding image.

B.2 Homophilic graph datasets

The *cora* dataset (Sen et al. 2008) is the citation network. Each node represents a paper, and each edge connects two nodes whose one of the corresponding papers cites another one. The node features are bag-of-word features derived from the corresponding paper. The node label is the academic category of the corresponding paper.

Table 1: Statistics of datasets leveraged in this work.

Datasets	# of nodes	# of edges	# of features	# of classes	Node homophily	Edge homophily
Cora	2,708	5,278	1,433	7	0.810	0.825
Citeseer	3,327	4,552	3,703	6	0.736	0.717
Pubmed	19,717	44,324	500	3	0.802	0.792
Photo	7,650	119,081	745	8	0.827	0.849
Computers	13,752	245,861	767	10	0.777	0.802
Arxiv	169,343	1,157,799	128	40	0.654	0.635
Chameleon	2,277	31,371	2,325	5	0.230	0.247
Squirrel	5,201	198,353	2,089	5	0.222	0.217
Actor	7,600	26,659	932	5	0.217	0.220
Cornell	183	277	1,703	5	0.122	0.111
Wisconsin	251	450	1,703	5	0.178	0.155
Texas	183	279	1,703	5	0.061	0.057
Penn94	41,554	1,362,229	4,814	3	0.470	0.483
Flickr	89,250	449,878	500	7	0.319	0.322

The *citeseer* dataset (Sen et al. 2008) is the citation network. Each node represents a paper, and each edge connects two nodes whose one of the corresponding papers cites another one. The node features are bag-of-word features derived from the corresponding paper. The node label is the academic category of the corresponding paper.

The *pubmed* dataset (Namata et al. 2012) is the citation network. Each node represents a paper, and each edge connects two nodes whose one of the corresponding papers cites another one. The node features are bag-of-word features derived from the corresponding paper. The node label is the academic category of the corresponding paper.

The *computer* dataset (Shchur et al. 2018) is the co-purchase network from McAuley et al. (2015). Each node represents a good, and each edge connects two nodes whose corresponding goods are bought together. The node features are bag-of-word features derived from the corresponding good's review. The node label is the category of the corresponding good.

The *photo* dataset (Shchur et al. 2018) is the co-purchase network from McAuley et al. (2015). Each node represents a good, and each edge connects two nodes whose corresponding goods are bought together. The node features are bag-of-word features derived from the corresponding good's review. The node label is the category of the corresponding good.

The *Arxiv* dataset (Hu et al. 2020) is the citation network. Each node represents a paper, and each edge connects two nodes whose one of the corresponding papers cites another one. The node features are Word2Vec (Mikolov et al. 2013) features derived from the corresponding paper. The node label is the academic category of the corresponding paper.

B.3 Dataset splits

In this section, we provide detailed descriptions regarding the training/validation/test splits of the datasets leveraged in our work. Specifically, for Squirrel, Actor, Wisconsin, Cornell, Texas, Chameleon, Penn94, Flickr, Cora, Citeseer, and Pubmed, we use the splits provided by PyG. For the Photo and Computers datasets, we adopt the split strategy suggested by Shchur et al. (2018), who originally proposed these datasets. Also, for the Arxiv dataset, we adopt the split strategy suggested by Hu et al. (2020), who originally proposed these datasets. Below, we report a detailed ratio for each dataset's training/validation/test splits.

Squirrel. We use 2,496/1,664/1,041 nodes for training/validation/test nodes with 10 different splits.

Actor. We use 3,648/2,432/1,520 nodes for training/validation/test nodes with 10 different splits.

Wisconsin. We use 120/80/51 nodes for training/validation/test nodes with 10 different splits.

Cornell. We use 87/59/37 nodes for training/validation/test nodes with 10 different splits.

Texas. We use 87/59/37 nodes for training/validation/test nodes with 10 different splits.

Chameleon. We use 1,092/729/456 nodes for training/validation/test nodes with 10 different splits.

Penn94. We use 91,445/30,481/47,417 nodes for training/-validation/test nodes with 5 different splits.

Flickr. We use 19,407/9,703/9,703 nodes for training/validation/test nodes with a single split.

Cora. We use 140/500/1,000 nodes for training/validation/test nodes with a single split.

Citeseer. We use 120/500/1,000 nodes for training/validation/test nodes with a single split.

Pubmed. We use 60/500/1,000 nodes for training/validation/test nodes with a single split.

Arxiv. We use 91,445/30,481/47,417 nodes for training/validation/test nodes with 10 different splits.

Photo. We use 765/765/6,120 nodes for training/validation/test nodes with 10 different splits.

Computers. We use 1,375/1,375/11,002 nodes for training/-validation/test nodes with 10 different splits.

C Experimental details

In this section, we provide detailed descriptions of the experiments conducted in this paper.

C.1 Machines and implementation

All experiments of this work are conducted on a machine with eight NVIDIA RTX 8000 D6 GPUs (48GB VRAM) and two Intel Xeon Silver 4214R processors. For each baseline method, we use the respective official code. For those whose official code is not available (i.e., NeCo and HLCL), we implement them according to their publications. FUEL is implemented based on Pytorch (Paszke et al. 2019) and PyG (Fey and Lenssen 2019). We train all the models using the Adam optimizer (Kingma and Ba 2015).

C.2 Hyperparameter configurations

Baselines. For the baseline methods, we follow the hyperparameter configurations provided in their publication or Github. For those whose hyperparameter configurations are not provided, we tuned their (1) learning rates, (2) hidden dimensions, (3) learning epochs, (4) dropout ratio, and (5) weight decay. Hyperparameter search spaces of the five components are as:

- Learning rate is tuned within $\{10^{-4}, 5\times 10^{-4}, 10^{-3}, 2\times 10^{-3}\}$.
- Embedding dimension is tuned within $\{256, 512, 1024\}$.
- Training epoch is tuned within $\{100, 200, 300, \cdots, 1000\}$.
- Dropout ratio is tuned within $\{0.1, 0.3, 0.5\}$.
- Weight decay is tuned within $\{10^{-1}, \cdots, 10^{-6}, 0\}$.

For those whose configurations are not provided and unique to each method, we fix them as their default settings. Lastly, for those whose default settings are not provided, we tuned the corresponding hyperparameters as below:

- For NeCo, we tuned the temperature hyperparameter τ within $\{0.1, 1.0\}$.
- For HLCL, we tuned the discrimination threshold k within $\{0.1, 0.2, 0.5, 0.9\}$.

The proposed method, FUEL. For the common hyperparameters (i.e., learning rates, training epochs, dropout ratios, and weight decays), FUEL shares the same hyperparameter search space with the baseline methods. For hyperparameters that are unique to FUEL, we tune each hyperparameter within the below search spaces:

- Distance loss weight in Step 1 (i.e., λ) is tuned within $\{0.01, 0.1, 1.0\}$.
- Number of nearest neighbors in Step 2 (i.e., N) is tuned within $\{5, 10, 15, 20\}$.
- Temperature hyperparameter in Step 2 (i.e., τ) is tuned within $\{3\times 10^{-2}, 2\times 10^{-2}, 1, 2, 10\}$.

The validation-best hyperparameter configuration of FUEL in each dataset is provided in our **?**.

Downstream task models. To perform the downstream tasks (i.e., node classification and clustering), we use the following models shared across all the unsupervised representation learning methods for evaluation. For node classification of small- and medium-scale graphs (i.e., $|\mathcal{V}| < 4\times 10^4$), we use a single-layer MLP classifier, having 256 hidden dimensions and a dropout ratio of 0.3. The classifier is trained for 500 epochs with 300 iteration patience for early stopping. For node classification of large-scale graphs (i.e., $|\mathcal{V}| \geq 4\times 10^4$), we use a single-layer MLP classifier, having 1024 hidden dimensions and a dropout ratio of 0.3. The classifier is trained for 1000 epochs with 300 iteration patience for early stopping. Learning rate and weight decay are fixed as 10^{-3} and 10^{-6}, respectively, for all the graphs. For clustering, we use the K-Means algorithm by setting K as the number of classes (i.e., the number of unique labels in the corresponding dataset).

C.3 Variants of FUEL

In this section, we provide details regarding the variants of FUEL we leverage in Section 5.5 of the main paper.

- **Variant that does not use the specialized clustering scheme (w/o Step 1).** This variant does not use the adaptive graph convolution model and uses the mean of $\mathbf{X}$, $\hat{\mathbf{A}}\mathbf{X}$. In other words, it uses $\hat{\mathbf{A}^2}\mathbf{X}$ as the intermediate node embeddings (i.e., $\mathbf{H} = \frac{1}{3}(\mathbf{X} + \hat{\mathbf{A}}\mathbf{X} + \hat{\mathbf{A}^2}\mathbf{X})$).
- **Variant that does not perform the refinement process (w/o Step 2).** This variant does not improve latent-class separability with nearest-neighbor pulling. In other words, it treats the intermediate node embeddings as the final node embeddings (i.e., $\mathbf{Z} = \mathbf{H}$).
- **Variant that does not use the skip connection for the refinement model (w/o SK).** This variant regards the output of a feed-forward neural network as the final node embeddings (i.e., $\mathbf{Z} = f_\theta(\mathbf{H})$).
- **Variant that does not use an exponential function for the distance loss (w/o Exp).** This variant does not use an exponential function for the distance loss. In other words, it leverages Eq. (18) for the distance loss:

$$\mathcal{L}_{dist} = \sum_{v_i, v_j \in \mathcal{V}'_+} \frac{d(\mathbf{z}_i, \mathbf{z}_j)}{\tau|\mathcal{V}'_+|} - \sum_{v_k, v_\ell \in \mathcal{V}'_-} \frac{d(\mathbf{z}_k, \mathbf{z}_\ell)}{\tau|\mathcal{V}'_-|}. \quad (18)$$

D Additional analysis

D.1 Analysis of existing baseline methods

In this section, we empirically analyze the accuracy of existing unsupervised node representation learning methods in estimating homophilic edges from a given graph. To this end, we analyze GREET (Liu et al. 2023) and HLCL (Yang and Mirzasoleiman 2024), where GREET is the strongest competitor.

Settings. We measure the homophily estimation accuracy of each method, which is defined as $\frac{|\mathcal{E}^+ \cap \hat{\mathcal{E}^+}|}{|\mathcal{E}^+|}$, where $\mathcal{E}^+ \subseteq \mathcal{E}$ is a subset of the edge set consist of homophilic edges and $\hat{\mathcal{E}^+}$ is also a subset of the sedge set which consist of the estimated homophilic edges by a particular model. We use the trained parameters that give the best validation accuracy on the node classification task.

Results. As shown in Table 2, both methods outperform the random guessing baseline marginally, at most just 6.1% performance gain. This result demonstrates that the homophily estimation techniques of these methods are inaccurate, implying the practical challenge of these approaches.

Table 2: **Homophilic edge estimation accuracy.** We report the estimation accuracy with the respective improvement (%) compared to random guessing.

	Cora	Citeseer	Chameleon	Squirrel	Actor
Random guessing	81.0	73.6	23.0	22.0	21.7
GREET	82.1 (+1.4%)	74.4 (+1.1%)	24.1 (+5.0%)	22.8 (+3.6%)	22.1 (+1.8%)
HLCL	81.6 (+0.7%)	73.7 (+0.1%)	24.4 (+6.1%)	23.1 (+5.0%)	21.8 (+0.5%)

D.2 About Calinski–Harabasz index

Measure details. The Calinski–Harabasz index (Caliński and Harabasz 1974), which is also known as the variance ratio criterion, is an unsupervised evaluation metric for the clustering algorithm. For a node $v_i \in \mathcal{V}$, we denote its feature as $\mathbf{x}_i \in \mathbb{R}^d$. Node are split into disjoint clusters $\mathcal{C}_1, \mathcal{C}_2, \cdots \mathcal{C}_c \subset \mathcal{V}$. Here, the Calinski–Harabasz index is defined as follows:

$$\frac{\frac{1}{c-1}\sum_{k\in[c]} |\mathcal{C}_k| \times \|\left(\sum_{v_t\in\mathcal{C}_k} \frac{\mathbf{x}_t}{|\mathcal{C}_k|}\right) - \left(\sum_{v_s\in\mathcal{V}} \frac{\mathbf{x}_s}{|\mathcal{V}|}\right)\|_2^2}{\frac{1}{n-c}\sum_{k\in[c]}\sum_{v_i\in\mathcal{C}_k} \|\mathbf{x}_i - \left(\sum_{v_t\in\mathcal{C}_k} \frac{\mathbf{x}_t}{|\mathcal{C}_k|}\right)\|_2^2}, \tag{19}$$

where the numerator corresponds to the mean inter-cluster distances and the denominator corresponds to the mean intra-cluster distances. Typically, when the data distribution within with-cluster is cohesive, and the data distribution of between clusters is distinctive, the index increases.

Application to our setting. In Section 3.2 of the main paper, we leverage the Calinski–Harabasz index as the latent-class separability measure of each embedding. To this end, we first run the K-Means clustering algorithm on each embedding, by setting K as the number of classes (i.e., the number of unique labels in the corresponding dataset). Then, by using each embedding and the clustering result of the corresponding embedding, we compute the Calinski–Harabasz index, by using the implementation provided by scikit-learn (https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.calinski_harabasz_score.html).

D.3 Advantages of exponential function in our distance loss

Because we consider the distance between every node pair, some pairs may have extremely large distances. If we use the arithmetic mean of all distances, these large values dominate the learning process, as their gradients are simply averaged. Consequently, the influence of other terms becomes negligible. We address this issue by employing an exponential function, which is a special case of the geometric average.

Note the partial derivative of our distance loss w.r.t. $\mathbf{z}_i$:

$$\frac{\partial \mathcal{L}_{dist}}{\partial \mathbf{z}_i} = \exp\left(\sum_{v_i,v_j\in\mathcal{V}'_+} \frac{d(\mathbf{z}_i,\mathbf{z}_j)}{|\mathcal{V}'_+|} - \sum_{v_k,v_\ell\in\mathcal{V}'_-} \frac{d(\mathbf{z}_k,\mathbf{z}_\ell)}{|\mathcal{V}'_-|}\right) \tag{20}$$

$$\times \left(\sum_{v_i,v_j\in\mathcal{V}'_+} \frac{d(\mathbf{z}_i,\mathbf{z}_j)}{|\mathcal{V}'_+|} - \sum_{v_k,v_\ell\in\mathcal{V}'_-} \frac{d(\mathbf{z}_k,\mathbf{z}_\ell)}{|\mathcal{V}'_-|}\right). \tag{21}$$

Since the global loss affects the gradient of each node embedding, when the overall loss is small, the impact of individual terms is diminished. Consequently, even if a particular term has a large loss value, its gradient contribution remains small under a low global loss. This mitigates outlier issues in the gradients and stabilizes the learning process.

D.4 Complexity analysis

In this section, we present the forward pass complexity analysis of FUEL for the number of nodes and number of edges. We first compute the complexity of Step 1. Graph-convolution-based encoding takes the complexity of $O(|\mathcal{V}| + |\mathcal{E}|)$ (Chiang et al. 2019). We conduct this for 2-hop, which is still bounded by $O(|\mathcal{V}| + |\mathcal{E}|)$. Then, computing clustering losses contain the all-node pair computation, which results in the complexity of $O(|\mathcal{V}|^2)$. Therefore, the complexity for the first step is equivalent to $O(|\mathcal{V}|^2)$.

We now compute the complexity for the second step. Encoding with a feed-forward network causes the complexity of $O(|\mathcal{V}|)$. Then, the distance loss computation contains the all-node pair distance computation, which has the complexity of $O(|\mathcal{V}|^2)$. Therefore, the complexity for the second step is equivalent to $O(|\mathcal{V}|^2)$.

Bringing Step 1 and Step 2 together, the resulting computation complexity of FUEL at the forward pass is equivalent to $O(|\mathcal{V}|^2) + O(|\mathcal{V}|^2) \equiv O(|\mathcal{V}|^2)$.

D.5 An extension of FUEL to large-scale graphs

In large-scale graphs, which contain many nodes and edges, it is typically hard to afford a heavy computation with respect to the number of nodes and edges, such as all-node-pairwise computation, which has the complexity of $O(|\mathcal{V}|^2)$. Therefore, we present a scalable version of FUEL-S, where the heavy computation taking the complexity of $O(|\mathcal{V}|^2)$ is removed. Notable differences between FUEL and FUEL-S are as follows. First, for the clustering loss, FUEL-S only leverages Eq. (3) and Eq. (4), omitting Eq. (5) (these equations indicate those in the main paper). Second, for the distance loss, FUEL does not use the entire $\mathcal{V}'_-$ as its negative samples, but only using its subset. Here, negative samples are sampled uniformly at random at every epoch. Lastly, upon Eq. (2) (of the main paper), we add a feed-forward neural network to obtain intermediate embeddings. Formally, we derive the intermediate embeddings as follows: $\mathbf{X}^* = g_\phi((\alpha_0\mathbf{I} + \alpha_1\tilde{\mathbf{A}} + \alpha_2\tilde{\mathbf{A}}^2)\mathbf{X}) \in \mathbb{R}^{|\mathcal{V}|\times d'}$, where g_ϕ is the added feed-forward neural network. This is because the availability of large-scale data enables the training of more expressive models without overfitting specific data points.

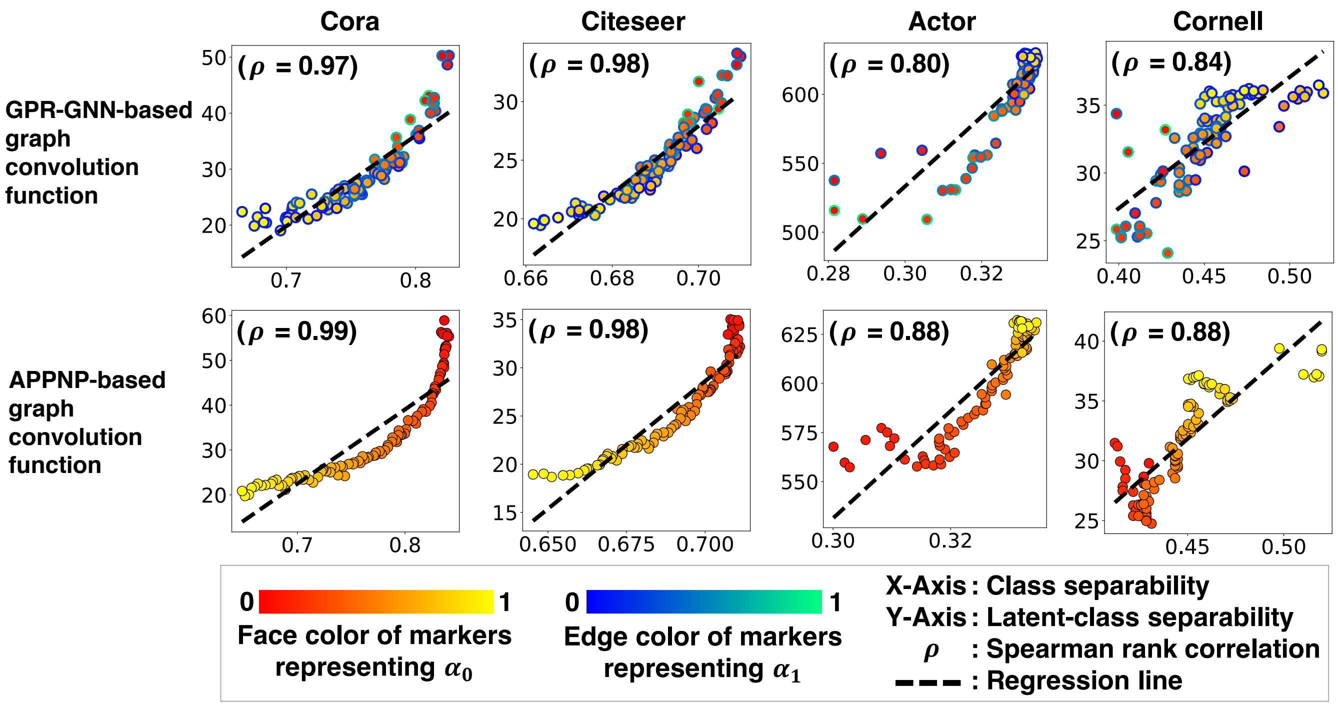

Figure 1: **Effectiveness of latent-class separability as a proxy for class separability.** Varying the graph convolution coefficients induces different degrees of graph convolution usage in the embeddings, which in turn leads to varying class separability. Notably, latent-class separability, our proposed proxy, strongly correlates with the actual class separability. This finding holds in both graph convolution functions (i.e., GPR-GNN-based and APPNP-based functions)

Therefore, the parameter ϕ is trained with cluster centroids and weights via clustering.

D.6 Analysis under other graph convolution functions

Recall that in Section 3.2 of the main paper, we use the following graph convolution function:

$$\mathbf{Z} = \alpha_0\mathbf{X} + \alpha_1\hat{\mathbf{A}}\mathbf{X} + \alpha_2\hat{\mathbf{A}^2}\mathbf{X}, \tag{22}$$

where $\hat{\mathbf{A}}$ is a degree-based row-wise normalized adjacency matrix (i. e., $\hat{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a diagonal adjacency matrix such that $\mathbf{D}_{ii}, \forall v_i \in \mathcal{V}$ is filled with the node degree of v_i). In this section, we conduct the same analysis as in Section 3.2 while using other graph convolution functions, which are APPNP (Gasteiger, Bojchevski, and Günnemann 2019) and GPR-GNN (Chien et al. 2021).

Analysis setting. For the APPNP-based graph convolution function, we use the following functional form:

$$\mathbf{Z} = \alpha_0^2\mathbf{X} + \alpha_0(1-\alpha_0)\tilde{\mathbf{A}}\mathbf{X} + (1-\alpha_0)^2\tilde{\mathbf{A}^2}\mathbf{X}, \tag{23}$$

where $\tilde{\mathbf{A}}$ is an adjacency matrix normalized via renormalization trick (Kipf and Welling 2017) (i.e., $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\bar{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is a diagonal degree matrix of $\bar{\mathbf{A}}$). For the GPR-GNN-based graph convolution function, we use the same functional form as Eq. (22), while we replace $\hat{\mathbf{A}}$ with $\tilde{\mathbf{A}}$. Other settings are the same as in Section 3.2.

Analysis result. As shown in Figure 1, still in other graph convolution functions, including APPNP and GPR-GNN, a high correlation between class separability and latent-class separability holds in both homophilic and heterophilic datasets. This result demonstrates that the effectiveness of latent-class separability as a proxy of class separability is not restricted to a particular graph convolution function, but generalizes to other graph convolution functions as well.

D.7 Node classification with a linear classifier

Recall that an MLP classifier has been used as a downstream node classifier in Section 5.2 of the main paper. In this section, we evaluate the effectiveness of FUEL in node classification with a linear classifier. To this end, we use a logistic classifier as a downstream node classifier and the three strongest baseline methods GraphMAE, GREET, and HeterGCL as the competitors of FUEL. Other experimental settings are the same as in Section 5.2 of the main paper. As shown in Table 3, FUEL outperforms all three strongest baseline methods in all the datasets. This result demonstrates that the effectiveness of FUEL is not limited to an MLP, but also generalizes well to linear classifiers.

D.8 Clustering evaluation with ARI

Recall that the Normalized Mutual Information (NMI) score has been used as an evaluation metric for clustering in Section 5.3 of the main paper. In this section, we use the Adjusted Rand Index (ARI) score as the clustering evaluation metric.

Table 3: **Node classification performance under linear classifier.** Mean and standard deviation of test accuracy values (×100) in the node classification task are reported. The best performances are highlighted in green color. FUEL achieves the best node classification performance in all the datasets.

	Cora	Citeseer	Photo	Chameleon	Squirrel	Actor
GraphMAE	82.5 (1.0)	72.0 (0.9)	92.2 (0.5)	45.6 (1.9)	37.0 (2.1)	28.2 (1.7)
GREET	82.8 (0.7)	71.5 (0.8)	91.9 (0.5)	57.6 (2.2)	40.6 (0.8)	36.3 (0.8)
HeterGCL	81.6 (1.5)	71.0 (0.5)	92.0 (0.6)	52.8 (2.8)	37.7 (1.6)	35.2 (1.3)
FUEL	83.1 (0.9)	73.0 (0.7)	93.8 (0.3)	71.9 (1.4)	58.7 (1.4)	37.4 (1.2)

Table 4: **Clustering performance measured with ARI.** Mean and standard deviation of ARI (Adjusted Rand Index) values (×100) in the node clustering task are reported. 0.0* indicates a value smaller than 10^{-4}. The best performances are highlighted in green color. FUEL achieves the best ARI clustering performance in 5 out of 6 datasets.

	Cora	Citeseer	Photo	Chameleon	Squirrel	Actor
GraphMAE	44.9 (2.2)	44.2 (1.0)	34.3 (5.4)	11.0 (0.6)	4.0 (0.2)	1.0 (0.0*)
GREET	52.4 (1.4)	44.6 (0.5)	44.9 (0.3)	12.4 (0.3)	3.8 (0.2)	4.2 (0.5)
HeterGCL	41.0 (4.5)	39.2 (2.5)	38.0 (0.8)	9.2 (0.2)	1.4 (0.0*)	3.8 (0.4)
FUEL	54.1 (0.5)	48.1 (0.7)	61.7 (0.3)	12.1 (1.2)	6.4 (0.1)	4.8 (0.2)

Table 5: **Node classification performance under new dataset versions.** Mean and standard deviation of test accuracy values (×100) in the node classification task are reported. FUEL achieves the second-best performance in these modified datasets.

	Modified chameleon	Modified Squirrel
GraphMAE	44.6 (3.7)	42.5 (2.1)
GREET	41.7 (2.6)	39.8 (2.1)
HeterGCL	43.2 (6.1)	40.5 (1.9)
FUEL	43.2 (3.4)	41.5 (1.2)

We compare FUEL with the three strongest baseline methods GraphMAE, GREET, and HeterGCL. Other experimental settings are the same as in Section 5.3 of the main paper. As shown in Table 4, FUEL outperforms the baseline methods in 5 out of 6 datasets. This result shows that the clustering effectiveness of FUEL is not limited to a particular evaluation metric, but also holds in another evaluation metric.

D.9 Evaluation using a different version of the datasets

For the Chameleon and Squirrel datasets, there are other versions of the datasets (Platonov et al. 2023). We conduct node classification evaluation for FUEL and the best three competitors (GraphMAE, GREET, and HeterGCL) in these datasets. Other experimental settings are the same as in Section 5.2 of the main paper. As shown in Table 5, FUEL achieves the second-best performance in these new datasets.

D.10 Evaluation using imbalanced graph datasets

In this section, we evaluate the effectiveness of FUEL in highly imbalanced multi-class graph datasets: LastFMAsia (Rozemberczki and Sarkar 2020). We use the two strongest baseline methods: HeterGCL and GREET. As shown in Table 6, FUEL outperforms the two strongest baseline methods, demonstrating its effectiveness in node classification for class-imbalanced graph.

Table 6: **Node classification performance under class-imbalanced graph.** Mean and standard deviation of test F1-macro values (×100) in the node classification task are reported. FUEL achieves the best performance in the LastFMAsia dataset, which is highly imbalanced.

Method	LastFMAsia
GREET	62.6 (0.8)
HeterGCL	61.4 (0.8)
FUEL	69.1 (0.4)

Table 7: **Motivation analysis with fisher discriminant analysis.** Correlation between fisher discriminant analysis value (class separability) and Calinski–Harabasz index (latent-class separability) of embeddings used in Section 3.2 of the main paper. The correlations are higher than 0.75 in all the cases.

Metric	Cora	Citeseer	Actor	Cornell
Spearman	0.991	0.985	0.948	0.883
Pearson	0.994	0.988	0.884	0.753

D.11 Motivation analysis with FDA

Note that in Section 3.2 of our main paper, we investigate the correlation between latent-class separability and class separability. To this end, we measure class separability of embeddings by: (1) train a classifier on the embeddings and

Table 8: **Cluster count analysis.** Node classification accuracy across different cluster counts. STDEV denotes a standard deviation. Across different cluster counts, the standard deviation remains less than 0.002, demonstrating that FUEL is insensitive to the choice of cluster counts.

Cluster counts	4	5	6	7	8	STDEV
Cora	0.837	0.836	0.837	0.838	0.837	0.0006
Citeseer	0.739	0.740	0.741	0.738	0.739	0.0010
Chameleon	0.729	0.732	0.731	0.731	0.729	0.0012
Squirrel	0.651	0.652	0.653	0.650	0.651	0.0010
Actor	0.382	0.382	0.382	0.381	0.380	0.0008

Table 9: **Noisy feature analysis.** Mean and standard deviation of test F1-macro values ($\times 100$) in the node classification task are reported. The best performances are highlighted in green color. Under noisy node features, FUEL still achieves the best performance in four out of five datasets.

	Cora	Citeseer	Chameleon	Squirrel	Actor
GREET	77.1 (2.1)	68.5 (2.1)	57.2 (3.3)	36.7 (4.4)	31.5 (1.0)
HeterGCL	79.6 (0.1)	71.0 (1.1)	56.9 (2.5)	45.8 (2.6)	31.4 (0.8)
FUEL	80.5 (0.3)	72.4 (0.9)	69.7 (2.6)	65.1 (1.3)	28.9 (1.1)

(2) evaluate the test accuracy of the trained classifier, treating the test accuracy as a measure of class separability. In this subsection, we evaluate the class separability without training a certain model; we measure it directly from the label by using fisher discriminant analysis (FDA) measure (Fisher 1936), which does have any learnable component. As shown in Table 7, correlations are higher than 0.75 in all the settings, supporting the effectiveness of latent-class separability as a proxy of class separability.

D.12 Cluster count analysis

In this section, we examine the sensitivity of FUEL to the number of clusters. We vary the number of clusters in FUEL and evaluate the resulting node classification performance. As shown in Table 8, for cluster counts between 4 and 8, the standard deviation of classification accuracy is smaller than 0.002, indicating that FUEL is insensitive to the choice of the cluster count.

D.13 Noisy feature analysis

In this section, we explore how robust is FUEL against the node feature corruption. To this end, we corrupt the node feature matrix by adding Gaussian noise to the node feature. For comparison, we use the two strongest baselines: GREET and HeterGCL. As shown in Table 9, FUEL outperforms the baseline methods in 4 out of 5 datasets, demonstrating its robustness against the feature corruption.

E Limitations and future works

In this section, we discuss the potential limitations and future works of our research. First, one of the limitations of our study is that it does not encompass graph-level tasks, specifically the learning of effective representations for entire graphs. Unsupervised representation learning of graphs is an actively studying research area (Li et al. 2024a; Wen et al. 2024; Wang et al. 2023), and therefore, we believe an extension of our work to the graph-level field would be a promising direction.

A further limitation is that the proposed method cannot be readily adapted to all types of graph neural networks. Emerging studies have developed self-supervised representation learning methods applicable to diverse neural network architectures (Wang et al. 2024; Li et al. 2024b). Consequently, broadening the applicability of our method to encompass a wider variety of neural networks would be an advantageous avenue for future research.

References

Caliński, T.; and Harabasz, J. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1): 1–27.

Chiang, W.-L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; and Hsieh, C.-J. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive universal generalized pagerank graph neural network. In *ICLR*.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR workshop on representation learning on graphs and manifolds*.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188.

Gasteiger, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Li, H.; Wang, X.; Zhang, Z.; Chen, H.; Zhang, Z.; and Zhu, W. 2024a. Disentangled Graph Self-supervised Learning for Out-of-Distribution Generalization. In *ICML*.

Li, J.; Zhang, H.; Wu, R.; Zhu, Z.; Wang, B.; Meng, C.; Zheng, Z.; and Chen, L. 2024b. A graph is worth 1-bit spikes: When graph contrastive learning meets spiking neural networks. In *ICLR*.

Lim, D.; Hohne, F.; Li, X.; Huang, S. L.; Gupta, V.; Bhalerao, O.; and Lim, S. N. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *NeurIPS*.

Liu, Y.; Zheng, Y.; Zhang, D.; Lee, V. C.; and Pan, S. 2023. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *AAAI*.

McAuley, J.; Targett, C.; Shi, Q.; and Van Den Hengel, A. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NeurIPS*.

Namata, G.; London, B.; Getoor, L.; Huang, B.; and Edu, U. 2012. Query-driven active surveying for collective classification. In *ICML workshop on mining and learning with graphs*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-gcn: Geometric graph convolutional networks. In *ICLR*.

Platonov, O.; Kuznedelev, D.; Diskin, M.; Babenko, A.; and Prokhorenkova, L. 2023. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *ICLR*.

Rozemberczki, B.; Allen, C.; and Sarkar, R. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2): cnab014.

Rozemberczki, B.; and Sarkar, R. 2020. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *CIKM*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. In *NeurIPS workshop on relational representation learning*.

Tang, J.; Sun, J.; Wang, C.; and Yang, Z. 2009. Social influence analysis in large-scale networks. In *KDD*.

Wang, H.; Kaddour, J.; Liu, S.; Tang, J.; Lasenby, J.; and Liu, Q. 2023. Evaluating self-supervised learning for molecular graph embeddings. In *NeurIPS*.

Wang, Z.; Wang, X.; Deng, H.; Liu, N.; Pan, S.; and Hu, C. 2024. Uncovering the Redundancy in Graph Self-supervised Learning Models. In *NeurIPS*.

Wen, Q.; Ju, M.; Ouyang, Z.; Zhang, C.; and Ye, Y. 2024. From Coarse to Fine: Enable Comprehensive Graph Self-supervised Learning with Multi-granular Semantic Ensemble. In *ICML*.

Yang, W.; and Mirzasoleiman, B. 2024. Graph Contrastive Learning under Heterophily via Graph Filters. In *UAI*.

Zeng, H.; Zhou, H.; Srivastava, A.; Kannan, R.; and Prasanna, V. 2020. Graphsaint: Graph sampling based inductive learning method. In *ICLR*.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *NeurIPS*.