6. Appendix

6.1. Implementation Details

The model is implemented with PyTorch and each benchmark is run on a single GPU. For a given dataset, the node features are processed as follows. If the dataset contains initial node features, they are normalize using the standard score. If the dataset does not contains initial node features, but carries node labels, they are used as the initial node features (only in graph classification benchmarks). Otherwise, if the dataset has neither, the node features are initialized with node degrees. Diffusion and pair-wise distance matrices are computed once in preprocessing step using NetworkX and Numpy packages. The shortest pair-wise distance matrix is computed by Floyd-Warshall algorithm, inversed in an element-wise fashion, and row-normalized using softmax.

6.2. Hyper-Parameters

In out experiments, we fixed α=0.2 and t=5 for PPR and heat diffusion, respectively. We used grid search to choose the hyper-parameters from the following ranges. For graph classification, we chose the number of GCN layers, number of epochs, batch size, and the C parameter of the SVM from [2, 4, 8, 12], [10, 20, 40, 100], [32, 64, 128, 256], and [10^{-3}, 10^{-2}, ..., 10^{2}, 10^{3}], respectively. For node classification, we set the number of GCN layers and the number of epochs and to 1 and 2,000, respectively, and choose the batch size from [2, 4, 8]. We also use early stopping with a patience of 20. Finally, we set the size of hidden dimension of both node and graph representations to 512. The selected parameters are reported in Table 6.

6.3. Effect of Number of Views

Furthermore, we investigated whether increasing the number of views increases the performance on down-stream tasks, monotonically. We extended number of views to three by anchoring the main view on adjacency matrix and considering two diffusion matrices, i.e, PPR and heat, as other views. The results shown in Table 7 suggest that unlike visual representation learning, extending the views does not improve the performance on the down-stream tasks. We speculate this is because different diffusion matrices carry similar information about the graph structure and hence using more of them does not introduce useful signals.

6.4. Effect of Pooling

In addition to mentioned sum pooling, we tried mean pooling and two variants of DiffPool: (1) DiffPool-1 where we use a single layer of DiffPool to aggregate all node representations into a single graph representation, and (2) DiffPool-2 where we use two DiffPool layers to compute the intermediate representations. The first layer projects nodes into a set of clusters, i.e., motifs, where the number of clusters is set as 25% of the number of nodes before applying DiffPool, whereas the second layer projects the learned cluster representations into a single graph representations. It is noteworthy that to train the model with Diffpool layers, we jointly optimize the contrastive loss with an auxiliary link prediction loss and an entropy regularization. The results shown in Table 6 suggest that sum pooling outperforms other pooling layers in 6 out of 8 benchmarks ,i.e., 5 out of 5 graph classification and 1 out if 3 node classification benchmarks. Also, results suggest that mean pooling achieves better results in 2 out of 3 node classification benchmarks.

6.5. Effect of Encoder

We also investigated the effect of assigning a dedicated encoder for each view or sharing an encoder across views. The results in Table 6 show that using a dedicated encoder for each view consistently achieves better results across all benchmarks. When using shared encoder, we also randomly sampled 2 out 4 views for each training sample in each mini-batch and contrasted the representation. We observed that unlike visual domain, it degraded the performance.

6.6. Effect of Negative Samples

Considering the importance of the number of negative samples in contrastive learning, we investigated the effect of batch size on the mode performance. For graph classification benchmarks, we used batch sizes of [16, 32, 64, 128, 256] and for classification benchmarks we evaluated batch sizes of [2, 4, 8]. As shown in Figure 2, we observe that increasing the batch size slightly increases the performance on graph classification task but has an negligible effect on the node classification.

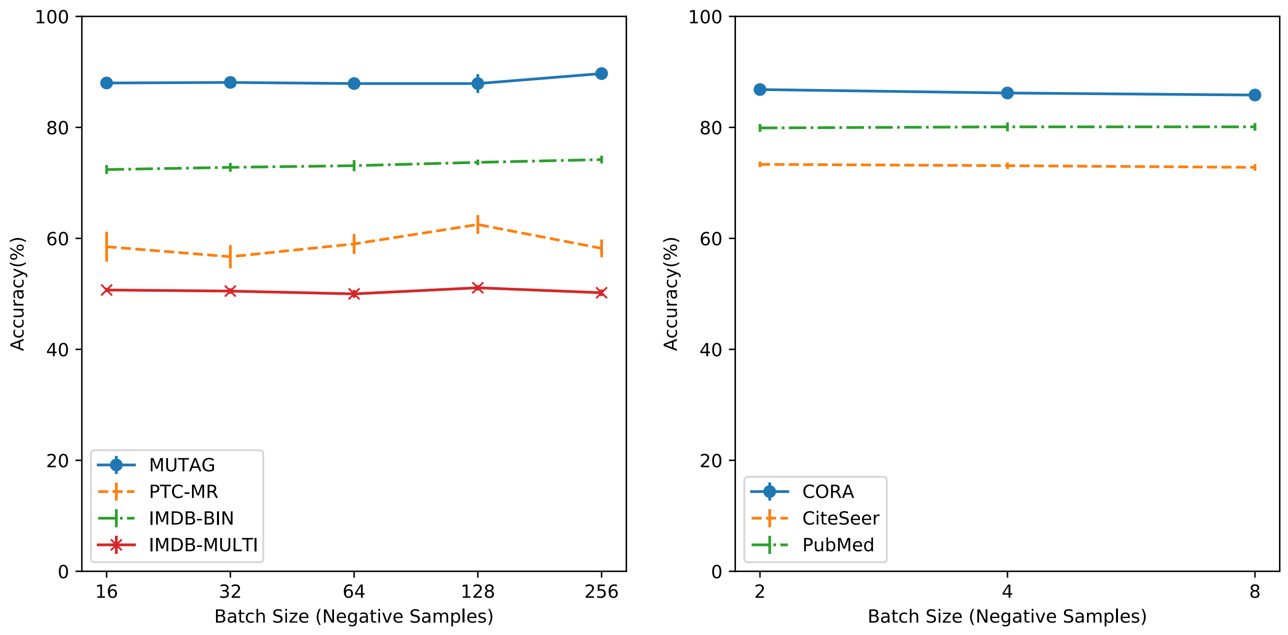

Figure 2. Effect of batch size (number of negative samples) on graph and node classification benchmarks. Increasing the batch size slightly increases the performance on graph classification task but has an negligible effect on the node classification.

Table 6. Summary of chosen hyper-parameters using grid search, and effect of graph pooling and encoders on the accuracy on both node and graph classification benchmarks. Using sum pooling and assigning dedicated encoders for each view achieve better results across the benchmarks.

		Node			*Graph*				
		CORA	CITESEER	PUBMED	MUTAG	PTC-MR	IMDB-BIN	IMDB-MULTI	REDDIT-BIN
HYPER	|LAYERS|	1	1	1	4	4	2	4	2
	|BATCHES|	2	2	2	20	40	20	40	20
	|EPOCHS|	2000	2000	2000	256	128	256	128	32
POOLING	SUM	86.2 ± 0.6	**73.3 ± 0.5**	79.6 ± 0.9	**89.7 ± 1.1**	**62.5 ± 1.7**	**74.2 ± 0.7**	**51.1 ± 0.5**	**84.5 ± 0.6**
	MEAN	**86.8 ± 0.5**	73.2 ± 0.6	**80.1 ± 0.7**	88.8 ± 0.7	60.9 ± 1.2	72.3 ± 0.5	49.1 ± 0.7	79.4 ± 0.5
	DIFFPOOL-1	84.6 ± 0.8	65.2 ± 1.7	76.1 ± 0.8	89.6 ± 0.9	61.1 ± 0.2	72.8 ± 0.6	49.4 ± 0.9	81.3 ± 0.4
	DIFFPOOL-2	83.2 ± 0.9	63.5 ± 1.5	75.7 ± 1.1	88.0 ± 0.8	56.6 ± 1.8	72.7 ± 0.4	50.6 ± 0.5	82.8 ± 0.6
ENC.	SHARED	86.2 ± 0.6	72.8 ± 0.6	79.5 ± 0.1	82.8 ± 1.9	55.9 ± 2.2	73.0 ± 0.7	50.0 ± 0.3	81.3 ± 2.0
	DEDICATED	**86.8 ± 0.5**	**73.3 ± 0.5**	**80.1 ± 0.7**	**89.7 ± 1.1**	**62.5 ± 1.7**	**74.2 ± 0.7**	**51.1 ± 0.5**	**84.5 ± 0.6**

Table 7. Effect of number of views on node classification accuracy.

#VIEWS	CORA	CITESEER	PUBMED
2	**86.8 ± 0.5**	**73.3 ± 0.5**	**80.1 ± 0.7**
3	85.3 ± 0.5	71.2 ± 0.7	79.9 ± 0.6