Supplementary of Graph Continual Learning with Debiased Lossless Memory Replay

Chaoxi Niu[a], **Guansong Pang**[b,*] **and Ling Chen**[a]

[a]AAII, University of Technology Sydney, Sydney, Australia
[b]Singapore Management University, Singapore

A Appendix

A.1 Different types of GCL Methods

Existing GCL methods can be roughly divided into three categories, i.e., regularization-based, parameter isolation-based, and data replay-based methods. Specifically, regularization-based methods preserve the parameters that are important to the previous tasks when learning new tasks via additional regularization terms. Parameter isolation-based methods maintain the performance on previous tasks by continually introducing new parameters for new tasks. Data replay-based methods employ a memory buffer to store representative data from previous tasks or a generative model to generate historical data. Then, the stored or generated historical data are replayed when learning new tasks.

A.2 Details on Datasets

- **CoraFull**[1]: It is a citation network containing 70 classes, where nodes represent papers and edges represent citation links between papers.
- **Arxiv**[2]: It is also a citation network between all Computer Science (CS) ARXIV papers indexed by MAG [2]. Each node in Arxiv denotes a CS paper and the edge between nodes represents a citation between them. The nodes are classified into 40 subject areas. The node features are computed as the average word-embedding of all words in the title and abstract.
- **Reddit**[3]: It encompasses Reddit posts generated in September 2014, with each post classified into distinct communities or subreddits. Specifically, nodes represent individual posts, and the edges between posts exist if a user has commented on both posts. Node features are derived from various attributes, including post title, content, comments, post score, and the number of comments.
- **Products**[4]: It is an Amazon product co-purchasing network, where nodes represent products sold in Amazon and the edges between nodes indicate that the products are purchased together. The node features are constructed with the dimensionality-reduced bag-of-words of the product descriptions.

* Corresponding Author (gspang@smu.edu.sg).
[1] https://docs.dgl.ai/en/1.1.x/generated/dgl.data.CoraFullDataset.html
[2] https://ogb.stanford.edu/docs/nodeprop/\#ogbn-arxiv
[3] https://docs.dgl.ai/en/1.1.x/generated/dgl.data.RedditDataset.html\#dgl.data.RedditDataset
[4] https://ogb.stanford.edu/docs/nodeprop/\#ogbn-products

A.3 More Implementation Details

All the continual learning methods including the proposed method are implemented based on the graph continual learning benchmark [4]. For ERGNN [7], the memory budget is set to be up to 800 per class to demonstrate the advantage of the proposed method. For SSM [5] and [6], we conduct experiments with the same memory budgets. However, to preserve the topological information, SSM and SEM need to store the neighbors of selected nodes. The number of neighbors for each node to store is set to 5 at each hop for both SSM and SEM.

Different GNN backbones, such as GCN [1] and SGC [3], can be applied to these continual learning methods. In the main paper, to have a fair comparison to the baselines [6], we employ a two-layer SGC model as the backbone. Specifically, the hidden dimension is set to 256 for all methods. The number of training epochs of each graph learning task is 200 with Adam as the optimizer and the learning rate is set to 0.005.

For the lossless memory learning in the proposed method, we employ the same two-layer SGC model as the GNN model to calculate the gradient-matching loss. The prototypical node representations are set as learnable parameters and are initialized with the original node attributes. The labels of the prototypical nodes are fixed during the learning process. Adam is used as the optimizer to learn the prototypical representations. The learning epochs and learning rate are set to 800 and 0.0001 for all tasks and datasets respectively. The scaling parameter τ in the debiased loss function is set to 1 for all experiments for simplicity.

The code is implemented with Pytorch (version: 1.10.0), DGL (version: 0.9.1), OGB (version: 1.3.6), and Python 3.8.5. Moreover, all experiments are conducted on a Linux server with an Intel CPU (Intel Xeon E-2288G 3.7GHz) and a Nvidia GPU (Quadro RTX 6000).

A.4 Memory budget efficiency

In the main paper, we report the results of the proposed method with different memory budgets under class-incremental learning. Here, we present the results under task-incremental learning in Figure 1. From the figure, we can see that all methods exhibit significantly improved performance under task-incremental learning compared to class-incremental learning. However, the learning-based methods (DeLoMe and CaT) are still more effective than the sampling-based method with tight memory budgets, indicating the importance of capturing the holistic semantics of the original graph into the memory

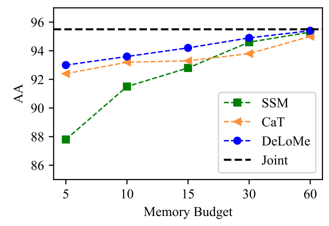
Table 1. Results under the class-incremental learning setting on four datasets with GCN as the backbone. Fine-tune and Joint are shown to respectively serve as approximated lower bound and upper bound performance. The best performance achieved by continual learning methods on each dataset is highlighted in bold. "↑" denotes the higher value represents better performance.

Methods	CoraFull		Arixv		Reddit		Products	
	AA/%↑	AF/%↑	AA/%↑	AF/%↑	AA/%↑	AF/%↑	AA/%↑	AF/%↑
Fine-tune	2.9±0.0	-94.7±0.1	4.9±0.0	-87.0±1.5	5.1±0.3	-94.5±2.5	3.4±0.8	-82.5±0.8
Joint	80.6±0.3	-	46.4±1.4	-	99.3±0.2	-	71.5±0.7	-
EWC	15.2±0.7	-81.1±1.0	4.9±0.0	-88.9±0.3	10.6±1.5	-92.9±1.6	3.3±1.2	-89.6±2.0
MAS	12.3±3.8	-83.7±4.1	4.9±0.0	-86.8±0.6	13.1±2.6	-35.2±3.5	15.0±2.1	-66.3±1.5
GEM	7.9±2.7	-84.8±2.7	4.8±0.5	-87.8±0.2	28.4±3.5	-71.9±4.2	5.5±0.7	-84.3±0.9
LwF	2.0±0.2	-95.0±0.2	4.9±0.0	-87.9±1.0	4.5±0.5	-82.1±1.0	3.1±0.8	-85.9±1.4
TWP	20.9±3.8	-73.3±4.1	4.9±0.0	-89.0±0.4	13.5±2.6	-89.7±2.7	3.0±0.7	-89.7±1.0
ERGNN	3.0±0.1	-93.8±0.5	30.3±1.5	-54.0±1.3	88.5±2.3	-10.8±2.4	24.5±1.9	-67.4±1.9
SSM-uniform	72.3±0.8	-15.5±1.5	45.1±1.1	-12.2±1.4	93.8±0.4	-2.0±0.3	61.8±1.2	-10.7±1.1
SSM-degree	74.4±0.2	-9.9±0.1	46.0±0.4	-11.3±0.8	94.0±0.2	-1.9±0.2	62.9±0.4	-10.3±0.5
SEM-curvature	79.6±0.3	-2.7±0.1	51.0±0.3	-6.7±0.6	95.1±0.6	-1.5±0.7	64.0±1.6	-11.2±1.8
CaT	79.5±0.4	-5.5±0.2	47.1±0.4	-13.7±0.2	98.0±0.1	-0.1±0.1	**70.6±1.0**	**-4.0±0.9**
DeLoMe (Ours)	**81.8±0.3**	**1.9±0.4**	**52.8±0.3**	**0.2±0.6**	**98.1±0.1**	**0.6±0.2**	67.0±0.8	-18.3±0.4

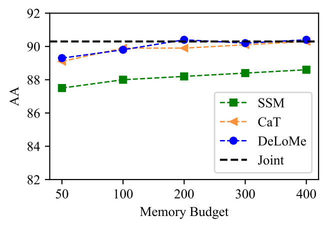
Table 2. Results under the task-incremental learning setting with GCN as the backbone.

Methods	CoraFull		Arixv		Reddit		Products	
	AA/%↑	AF/%↑	AA/%↑	AF/%↑	AA/%↑	AF/%↑	AA/%↑	AF/%↑
Fine-Tune	58.0±1.7	-38.4±1.8	61.7±3.8	-28.2±3.3	73.6±3.5	-26.9±3.5	67.6±1.6	-25.4±1.6
Joint	95.2±0.2	-	90.3±0.2	-	99.4±0.1	-	91.8±0.2	-
EWC	78.9±2.4	-15.5±2.3	78.8±2.7	-5.0±3.1	91.5±4.2	-8.1±4.6	90.1±0.3	-0.7±0.3
MAS	93.0±0.5	-0.6±0.2	88.4±0.2	-0.0±0.0	98.6±0.5	-0.1±0.1	91.2±0.6	-0.5±0.2
GEM	91.6±0.6	-1.8±0.6	87.3±0.6	2.8±0.3	91.6±5.6	-8.1±5.8	87.8±0.5	-2.9±0.5
LwF	56.1±2.0	-37.5±1.8	84.2±0.5	-3.7±0.6	80.9±4.3	-19.1±4.6	66.5±2.2	-26.3±2.3
TWP	92.2±0.5	-0.9±0.3	86.0±0.8	-2.8±0.8	87.4±3.8	-12.6±4.0	90.3±0.1	-0.5±0.1
ERGNN	90.6±0.1	-3.7±0.1	86.7±0.1	**11.4±0.9**	98.9±0.0	-0.1±0.1	89.0±0.4	-2.5±0.3
SSM-uniform	94.5±0.8	-0.1±0.9	88.8±1.2	-2.1±0.7	98.8±0.3	-0.9±0.6	90.9±2.8	-2.2±1.0
SSM-degree	94.3±1.1	0.9±0.5	88.1±1.3	-1.0±0.4	99.0±0.2	-0.4±0.2	91.1±0.9	-1.8±0.8
SEM-curvature	95.0±0.5	-0.2±0.8	88.8±0.1	-1.0±0.2	99.2±0.1	-0.1±0.3	91.6±0.5	-1.5±0.8
CaT	94.3±0.2	0.5±0.5	90.2±0.2	0.2±0.1	99.2±0.0	**0.1±0.1**	**94.3±0.6**	**0.1±0.2**
DeLoMe (Ours)	**95.2±0.3**	**1.3±0.1**	**90.6±0.3**	2.3±1.1	**99.3±0.1**	-0.2±0.1	94.2±0.1	-1.4±0.1

buffer. Note that the performance gain of DeLoMe over CaT is not as significant as observed in class-incremental learning, which is attributed to the fact that task-incremental learning is less challenging than class-incremental learning.



(a) CoraFull



(b) Arxiv

Figure 1. AA results with different memory budgets on CoraFull and Arxiv under the task-incremental learning.

A.5 *Results with GCN as Backbone*

As stated above, different GNNs can be used as the backbone of DeLoMe. In the main paper, we report the results with SGC [3] as the backbone. In this subsection, we further present the results with GCN [1] as the backbone. Specifically, we employ the same baselines as in the main paper for comparison, and the results under the class- and task-incremental learning are shown in Table 1 and Table 2 respectively. From the results, we can draw similar observations as in the main paper, i.e., the proposed DeLoMe can effectively overcome the catastrophic forgetting problem in graph continual learning by utilizing lossless memory and debiased memory replay. The results also verify the effectiveness of DeLoMe with different backbones.

References

[1] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[2] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[3] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[4] X. Zhang, D. Song, and D. Tao. Cglb: Benchmark tasks for continual graph learning. *Advances in Neural Information Processing Systems*, 35: 13006–13021, 2022.

[5] X. Zhang, D. Song, and D. Tao. Sparsified subgraph memory for continual graph representation learning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1335–1340. IEEE, 2022.

[6] X. Zhang, D. Song, D. Tao. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[7] F. Zhou and C. Cao. Overcoming catastrophic forgetting in graph neural

networks with experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4714–4722, 2021.