Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Supplementary materials for

Jianhao GUO, Zixuan NI, Yun ZHU, Siliang TANG, 2025. E-CGL: an efficient continual graph learner. *Front Inform Technol Electron Eng*, 26(8):1441-1453.
https://doi.org/10.1631/FITEE.2500162

1 Theoretical proof

1.1 Proof for equation 11

Based on the definition of Q and r in Eq.8 and Eq.10 respectively, we can derive that:

$$\begin{aligned}(Qr)_i &= \sum_{j\in\mathcal{V}} \frac{s_{ij}}{\sum_{k\in\mathcal{V}} s_{kj}} r_j \\ &= \sum_{j\in\mathcal{V}} \frac{s_{ij}}{\sum_{k\in\mathcal{V}} s_{kj}} \frac{\sum_{k\in\mathcal{V}} s_{jk}}{z} \\ &= \frac{1}{z}\sum_{j\in\mathcal{V}} s_{ij} \\ &= 1 \cdot r_i.\end{aligned} \tag{S1}$$

Therefore we have $\mathbf{1}\cdot r = Qr$.

1.2 Simplify r using Taylor expansion

Consider the r_i in Eq.10 being unnormalized: $\hat{r}_i = \sum_{j\in\mathcal{V}} s(i,j)$ and with the Radial Basis Function $s(i,j) = e^{-\gamma||\mathbf{x}_i-\mathbf{x}_j||_2^2}$ as similarity, it can be simplified using Taylor expansion:

$$\begin{aligned}\hat{r}_i =& \sum_{j\in\mathcal{V}} e^{-\gamma\|x_i-x_j\|_2^2} \\ =& \sum_{j\in\mathcal{V}} e^{-\gamma\left(\|x_i\|_2^2+\|x_j\|_2^2-2x_i^Tx_j\right)} \\ \approx& e^{-\gamma\|x_i\|_2^2}\sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2}\left(1+2\gamma x_i^Tx_j+\frac{1}{2}\left(2\gamma x_i^Tx_j\right)^2\right) \\ =& e^{-\gamma\|x_i\|_2^2}\left[\sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2} + x_i^T\left(2\gamma\sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2}x_j\right)\right. \\ & \left. + x_i^T\left(2\gamma^2\sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2}x_jx_j^T\right)x_i\right] \\ \equiv& w_i\left[a + x_i^Tb + x_i^TCx_i\right],\end{aligned} \tag{S2}$$

where $w_i = e^{-\gamma\|x_i\|_2^2}$, $a = \sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2}$, $b = 2\gamma\sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2}x_j$, and $c = 2\gamma^2\sum_{j\in\mathcal{V}} e^{-\gamma\|x_j\|_2^2}x_jx_j^T$ can be pre-calculated and all of them requires an $O(|\mathcal{V}|D^2)$ time complexity. Considering $|\mathcal{V}| \gg D$ in most cases, such cost is acceptable for large graphs.

2 Pseudo-code of E-CGL

The pseudo-code of E-CGL is presented in Algorithm 1.

Algorithm 1 Framework of E-CGL

Require: Memory bank: $\mathcal{M}$; Continual graphs: $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$; max epochs E
Ensure: Classification model f parameterized by $\mathbf{W}$

for $t = 1$ **to** T **do**
  /**Train**/
  Obtain current training set $G_t = (\mathcal{V}_t^{tr}, \mathcal{E}_t^{tr})$ and memory buffer $\mathcal{M}$
  for *epoch* $= 1$ **to** E **do**
    Compute loss function:
    $\mathcal{L}_{\text{new}} = \sum_{i \in \mathcal{V}_t^{tr}} L_{\text{CE}}(f_{\text{MLP}}(\mathbf{x}_i; \mathbf{W}_{\text{MLP}}), \mathbf{y}_i)$
    $\mathcal{L}_{\text{replay}} = \sum_{j \in \mathcal{M}} L_{\text{CE}}(f_{\text{MLP}}(\mathbf{x}_j; \mathbf{W}_{\text{MLP}}), \mathbf{y}_j)$
    $\mathcal{L} = \mathcal{L}_{\text{new}} + \lambda \mathcal{L}_{\text{replay}}$
    Update model parameters:
    $\mathbf{W}_{\text{MLP}} \leftarrow \text{argmin}_{\mathbf{W}_{\text{MLP}} \in \Theta} \mathcal{L}$
  end for
  Calculate importance rank π_{Imp} by Eq.12
  Calculate diversity rank π_{Div} by Eq.13
  Sample and update memory bank:
  $\mathcal{M} \leftarrow \mathcal{M} \cup \text{argtopk}\pi_{\text{Imp}} \cup \text{argtopk}\pi_{\text{Div}}$

  /**Inference**/
  Initialize f_{GCN} using $\mathbf{W}_{\text{MLP}}$
  for $tt = 1$ **to** t **do**
    Predict on testing data:
    $\hat{\mathbf{Y}}_{tt} = f_{\text{GCN}}(G_{tt}^{te}; \mathbf{W}_{\text{MLP}})$
  end for
end for

3 Implementation details

3.1 Running environment

The experiments were conducted on a machine with NVIDIA 3090 GPU (24GB memory). The E-CGL model and other baselines were implemented using Python 3.9.16[1], PyTorch 1.12.1[2], CUDA 11.3[3], and DGL 0.9.1[4]. The code was developed based on the benchmark CGLB.

3.2 Model configurations

For a fair comparison, a two-layer GCN with a hidden dimension of 256 is used as the backbone for all compared methods. Unless otherwise specified, the same training configurations, including optimizer, learning rate, weight decay, and training epochs, are used for all baseline methods. Specifically, a batch size of 8000 is used when batching is necessary. Adam is employed as the optimizer with a learning rate of 0.005, and the weight decay is set to 5×10^{-4}. Each task is trained for 200 epochs. The reported mean and

[1]https://www.python.org/downloads/release/python-3916/
[2]https://pytorch.org/get-started/previous-versions/
[3]https://developer.nvidia.com/cuda-11.3.0-download-archive
[4]https://www.dgl.ai/

standard deviations of AA and AF are based on five independent runs with random seeds ranging from 0 to 4.

Table. S1 Hyperparameter list for compared methods.

Methods	Hyperparameters
LwF	$\lambda_{\text{dist}} : \{1.0, 10.0\}$, T : $\{2.0, 20.0\}$
EWC	memory strength : $\{10000.0\}$
MAS	memory strength : $\{10000.0\}$
GEM	memory strength : $\{0.5\}$, memory nums : $\{100\}$
TWP	$\lambda_{\text{l}} : \{10000.0\}$, $\lambda_{\text{t}} : \{10000.0\}$, $\beta : \{0.01\}$
ER-GNN	sample budget : $\{500, 1000, 5000\}$, $d : \{0.5\}$, sampler : $\{\text{MF}, \text{CM}\}$
DyGRAIN	results are imported from their original paper
SSM	c_node budget : $\{100\}$, neighbor budget : $\{[0, 0]\}$, $\lambda : 1$
CaT	sample budget : $\{100, 1000, 3000, 5000\}$
GCL-SAGE	$\alpha : 0.5$, $\beta : 0.5$, buffer memory slots : $\{500\}$
TACO	reduction ratio : 0.5, memory buffer : 200
DSLR	$\beta : \{0.05, 0.1\}$, $\lambda : \{0.5\}$, $N : \{5\}$, $K : \{50\}$, $\tau : \{0.8\}$, $r : \{0.15, 0.2, 0.25, 0.3\}$, buffer size : $\{100, 200, 3000\}$
E-CGL	sample budget : $\{1000, 3000, 5000\}$, diversity ratio : $\{0.1, 0.25\}$, $\lambda : \{1.0\}$

3.3 Hyperparameters

Comprehensive hyperparameters specific to each method are provided in Table S1, and the reported results are based on the best outcomes obtained through grid search on these hyperparameters. To reproduce the results of E-CGL, set the sampling budget for Graph Dependent Replay as 1000 for CoraFull, 3000 for OGBN-Arxiv, 5000 for Reddit and OGBN-Products. Among all sampled nodes, 25% are selected using diversity sampling and 75% are selected using importance sampling. The loss weight λ is set to 1.

Table. S2 Statistics of the node classification datasets.

Datasets	CoraFull	OGBN-Arxiv	Reddit	OGBN-Products
#Nodes	19,793	169,343	227,853	2,449,028
#Edges	130,622	1,166,243	114,615,892	61,859,036
#Classes	70	40	40	46
#Tasks	14	10	10	23
#Cls Per Task	5	4	4	2

4 Additional experiments

4.1 Choice of inference encoders

As mentioned in Section 3.3, the networks used during the inference phase can be generalized to any message-passing-based GNN. In most experiments in this section, we adopt GCN as the inference model. Without loss of generality, we further evaluated other GNNs, including GraphSAGE, GAT, and GIN, as alternative inference models.

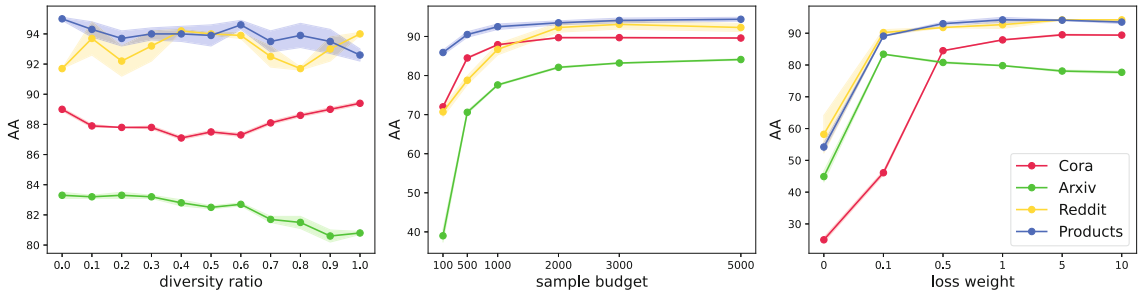
Specifically, when adapting GAT, we replaced the original single-layer feedforward neural network used for attention computation ($a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$) with cosine similarity and employ single-head attention. This modification avoids introducing additional model parameters and ensures parameter consistency between the MLP used during training and the GAT used during inference.

Table. S3 Results of E-CGL using different GNN encoders for inference under task-IL setting.

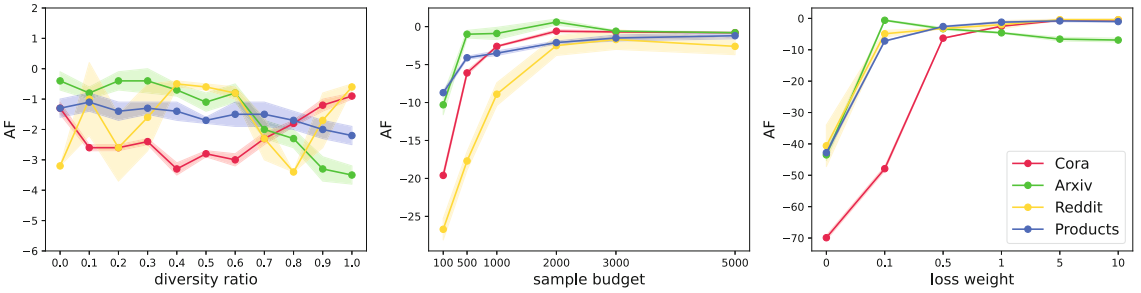
Inference GNN	CoraFull		OGBN-Arxiv		Reddit		OGBN-Products	
	AA/%↑	AF/%↑	AA/%↑	AF/%↑	AA/%↑	AF/%↑	AA/%↑	AF/%↑
GCN (default)	**89.6**±0.1	-2.5±0.2	82.1±1.0	0.2±0.2	92.2±0.7	-2.7±0.8	93.9±0.6	-1.2±0.3
GraphSAGE	88.2±0.4	-3.9±0.2	82.3±4.1	-5.3±0.7	91.3±0.1	-4.4±1.1	92.3±0.8	-2.0±0.4
GAT	89.5±0.1	-0.7±0.0	**83.0**±2.2	-0.1±0.5	92.8±0.9	-1.6±0.6	**94.2**±1.0	-0.7±0.1
GIN	88.3±0.1	-1.4±0.1	82.2±0.9	-3.0±2.5	**93.0**±1.2	-2.1±0.6	93.4±0.8	-1.1±0.4

The results are presented in Table S3. As shown, the differences in inference performance across different encoders are relatively small, as they all share the same model parameters and differ only in the message-passing mechanism. This supports the generalizability of E-CGL across various GNN architectures. Additionally, GAT performs slightly better than other encoders, suggesting that the attention mechanism may play a beneficial role in message propagation during inference.

4.2 Parameter sensitivity



(a) The y-axis is average accuracy (AA).



(b) The y-axis is average forgetting (AF).

Fig. S1 Parameter sensitivity analysis on E-CGL with the shallow shades showing the variances. Left: diversity sampling ratio. Middle: sampling budget for Graph Dependent Replay. Right: loss weight λ.

We conducted a parameter sensitivity analysis on three factors: *1) diversity sampling ratio*, *2) sampling budget for* $\mathcal{M}$, and *3) loss weight* λ. The average accuracy (AA) and the average forgetting (AF) results are depicted in Fig. S1.

Firstly, we observed that the diversity sampling ratio has a minimal impact on the results of continual graph learning. Both AA and AF exhibit slight fluctuations within a narrow range as the diversity ratio varies. The differing curve trends also indicate that the effect of different sampling strategies depends on specific datasets, highlighting the necessity of a combined strategy.

Secondly, the performance of E-CGL consistently improves with an increase in the sampling budget. This aligns with expectations, as a larger budget enables the replay of more nodes, effectively enhancing model performance. However, it is important to note that the budget cannot be infinitely expanded due to storage limitations and concerns regarding training efficiency. Eventually, when the sampling budget

becomes extremely large, the replay-based method converges to joint training.

Lastly, the model's performance initially increases and then decreases (more noticeably on OGBN-Arxiv) as the loss weight λ is raised. This pattern aligns with intuition. In our main experiment, we simply set λ to 1.

4.3 Visualization

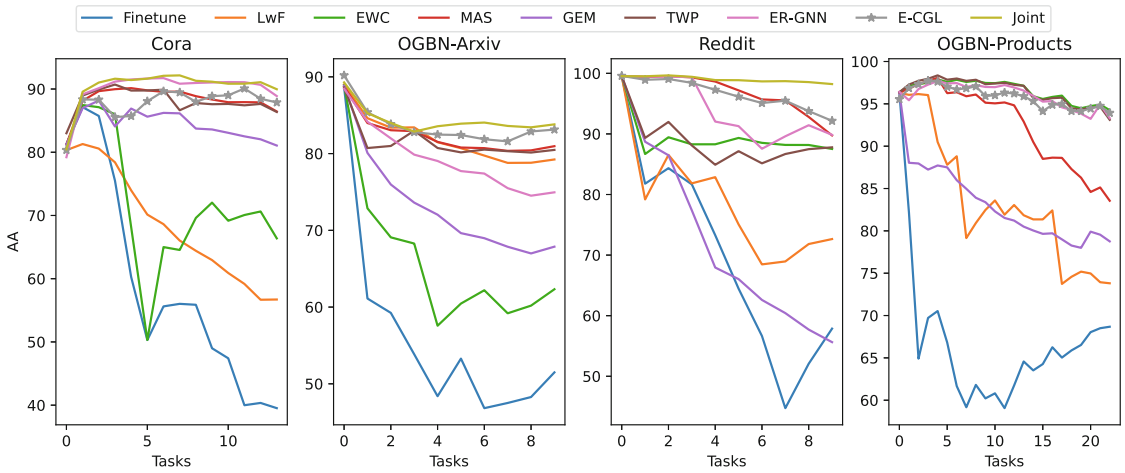


Fig. S2 Visualization: learning curves of AA over task sequences. Note: The curve for joint training on OGBN-Products is unavailable due to resource limitations.

Learning curve We visualized the learning curves of certain methods in Fig. S2 to provide further insight into the training process over the task sequence. The learning curve represents the trend of average accuracy (y-axis) as the number of tasks (x-axis) increases, reflecting the model's adaptability in the continual learning process.

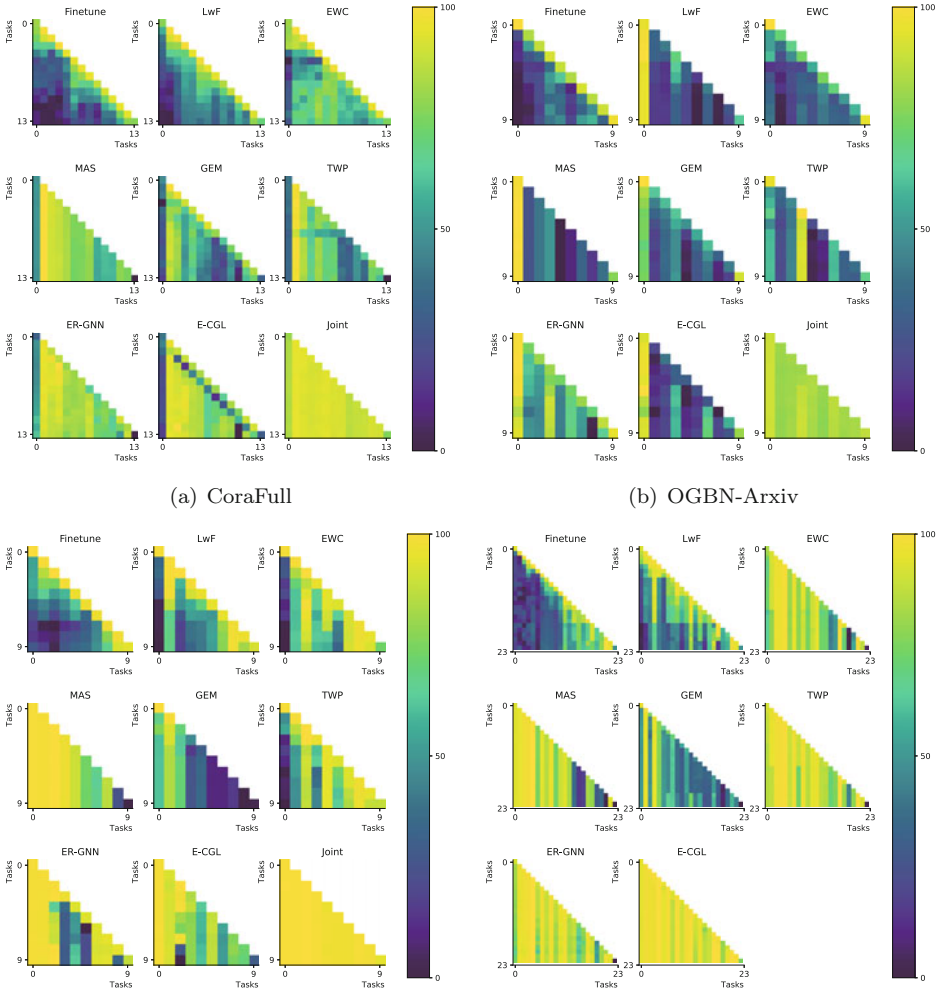
All methods exhibit a downward trend in average accuracy, indicating the presence of catastrophic forgetting. The rate of decline in the learning curves serves as an indicator of dataset difficulty, as more challenging datasets hinder the modelâĂŹs ability to adapt effectively. OGBN-Arxiv and Reddit, which exhibit steeper declines in AA values, are relatively challenging compared to the simpler OGBN-Products dataset.

Another important observation is that replay-based approaches, including ER-GNN and E-CGL, consistently outperform other methods. This finding suggests that the interdependencies in graph data are strong and need to be explicitly maintained to mitigate catastrophic forgetting. By leveraging past samples during training, replay-based methods are able to retain important information and sustain performance on previously learned tasks.

Performance Matrix We generated performance matrices to visualize the performance of several baseline methods and E-CGL under task-IL setting, as shown in Fig. S3. For the performance matrix, each color block $\mathbf{M}^{p}_{i,j}$ at position i-th row and j-th column represents the average accuracy (AA) on task j (where $j = 1, ..., i$) after the model has been trained on tasks 1 to i. The brightness of the color indicates the level of AA, with brighter colors corresponding to higher accuracy.

Similar to the analysis conducted in the Section 4.2, most methods fall between the lower bound of fine-tuning and the upper bound of joint training. Generally, E-CGL demonstrates higher values, and graph-specific continual learning techniques show better overall performance.

One interesting finding in the performance matrix is related to the diagonal entries, which represent the model's ability to adapt to new tasks. It can be observed that in certain cases, some regularization methods



(a) CoraFull (b) OGBN-Arxiv

(c) Reddit (d) OGBN-Products

Fig. S3 Visualization: performance matrices on CoraFull, OGBN-Arxiv, Reddit, and OGBN-Products.

(e.g., TWP on CoraFull, GEM on OGBN-Products) exhibit weaker adaptation ability. We speculate that these methods impose constraints on model parameter updates, which can preserve the model's performance on previous tasks but limit its ability to adapt to new tasks.

The overall value range of the performance matrix also directly reflects the difficulty of each dataset. It is noticeable that most methods have darker (i.e., lower) values on OGBN-Arxiv compared to OGBN-Products. This observation aligns with the findings in visualization of learning curve.