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

E-CGL: an efficient continual graph learner*#

Jianhao GUO, Zixuan NI, Yun ZHU, Siliang TANG‡

Digital Media Computing & Design Lab, College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

E-mail: guojianhao@zju.edu.cn; zixuan2i@zju.edu.cn; zhuyun_dcd@zju.edu.cn; siliang@zju.edu.cn

Received Mar. 14, 2025; Revision accepted June 3, 2025; Crosschecked July 10, 2025

Abstract: Continual learning (CL) has emerged as a crucial paradigm for learning from sequential data while retaining previous knowledge. Continual graph learning (CGL), characterized by dynamically evolving graphs from streaming data, presents distinct challenges that demand efficient algorithms to prevent catastrophic forgetting. The first challenge stems from the interdependencies between different graph data, in which previous graphs influence new data distributions. The second challenge is handling large graphs in an efficient manner. To address these challenges, we propose an efficient continual graph learner (E-CGL) in this paper. We address the interdependence issue by demonstrating the effectiveness of replay strategies and introducing a combined sampling approach that considers both node importance and diversity. To improve efficiency, E-CGL leverages a simple yet effective multilayer perceptron (MLP) model that shares weights with a graph neural network (GNN) during training, thereby accelerating computation by circumventing the expensive message-passing process. Our method achieves state-of-the-art results on four CGL datasets under two settings, while significantly lowering the catastrophic forgetting value to an average of -1.1%. Additionally, E-CGL achieves the training and inference speedup by an average of $15.83\times$ and $4.89\times$, respectively, across four datasets. These results indicate that E-CGL not only effectively manages correlations between different graph data during continual training but also enhances efficiency in large-scale CGL.

Key words: Graph neural networks; Continual learning; Dynamic graphs; Continual graph learning; Graph acceleration

https://doi.org/10.1631/FITEE.2500162 **CLC number:** TP181

1 Introduction

Graphs have garnered significant research attention due to their ubiquity in modeling relational data (Kipf and Welling, 2017; Veličković et al., 2018; Xu et al., 2019). In real-world applications, graph data continuously expand, and new patterns emerge over time. For instance, recommendation networks may introduce new product categories, citation networks may see the rise of novel research topics, and chemical design may discover new molecules and drugs. To adapt to these evolving scenarios and provide up-to-date predictions, graph models must constantly learn and update. However, traditional training strategies (Niepert et al., 2016; Kipf and Welling, 2017) experience catastrophic forgetting (Yuan et al., 2023) when adapting to new data, resulting in poor performance on previous tasks.

As a result, methods that can rapidly adapt to new classes while maintaining performance on previous tasks are required. Continual learning (CL, also known as incremental learning or lifelong learning) aims to do this by learning from new data while retaining prior knowledge (Thrun, 1994). While CL

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62272411), the Key R&D Projects in Zhejiang Province (Nos. 2024C01106 and 2025C01030), and the Zhejiang Natural Science Foundation (No. LRG25F020001)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2500162) contains supplementary materials, which are available to authorized users

ORCID: Jianhao GUO, https://orcid.org/0000-0002-4285-5328; Siliang TANG, https://orcid.org/0000-0002-7356-9711

has been extensively studied in areas such as computer vision (Rebuffi et al., 2017; Buzzega et al., 2020; Ni et al., 2023) and natural language processing (Srinivasan et al., 2022; Fan et al., 2022), its application to graph-structured data presents unique challenges for a diverse set of downstream tasks, including supervised node classification and unsupervised dynamic network embedding (Choi et al., 2024; Gao et al., 2024; Wang ZZ et al., 2025).

The first challenge arises from the topological interdependence of graph snapshots, in which new tasks inherit structural and attributive dependencies from previous tasks. Topological interdependence occurs when sequential graph snapshots $\{G_1, G_2, ..., G_T\}$ share nodes/edges or have attribute continuity, resulting in conditional dependencies between their distributions (Fig. 1). Specifically, in continual graph learning (CGL), the conditional likelihood of observing G_t depends on previous graphs:

$$p(\theta|G_{1:t}) \propto \frac{p(G_t|G_{1:t-1},\theta)p(\theta|G_{1:t-1})}{p(G_t)}, \quad (1)$$

where $p(\theta|G_{1:t})$ refers to the posterior distribution of the model parameter θ after observing all the snapshots of the graph from round 1 to round t, $p(\theta|G_{1:t-1})$ is an estimate of θ when only the first $t-1$ snapshots of the graph are observed, and is the "prior knowledge" or a priori of the model parameters prior to the start of the current task, $p(G_t)$ is a non-computable constant, referring to the probability of the entire graph appearing, and $p(G_t|G_{1:t-1},\theta)$ refers to the conditional probability of the occurrence of the current graph snapshot G_t, given the knowledge of the past graph snapshots G_1 to G_{t-1} and θ. Additionally, $p(G_t|G_{1:t-1},\theta)$ encodes dependencies from the prior structure and attributes, not just a priori static parameter. This interdependence is fundamentally different from image-based CL, where tasks are conditionally independent given model parameters. Such interdependence exacerbates catastrophic forgetting when previous snapshots are no longer accessible (Hu XT et al., 2021; Su et al., 2025).

The second challenge is the efficiency and scalability of handling increasingly large graphs. Real-world graphs (e.g., social networks) grow exponentially, necessitating efficient model updates. However, message-passing mechanisms in graph neural networks (GNNs) have $O(N^2)$ complexity (N is the number of nodes), hindering rapid adaptation to new trends, which is a critical requirement for applications such as real-time recommendation systems.

In this paper, we propose E-CGL, an efficient continual graph learner, designed for supervised node classification in evolving graphs. To model inter-task dependencies and mitigate catastrophic forgetting, we demonstrate the effectiveness of replay strategies and propose a combined sampling approach that considers both node importance and diversity. First, we design an importance sampling strategy that leverages an attributed-PageRank algorithm to incorporate both topology and attributes. Second, we introduce a diversity sampling strategy for capturing novel patterns by modeling differences between nodes and their neighbors.

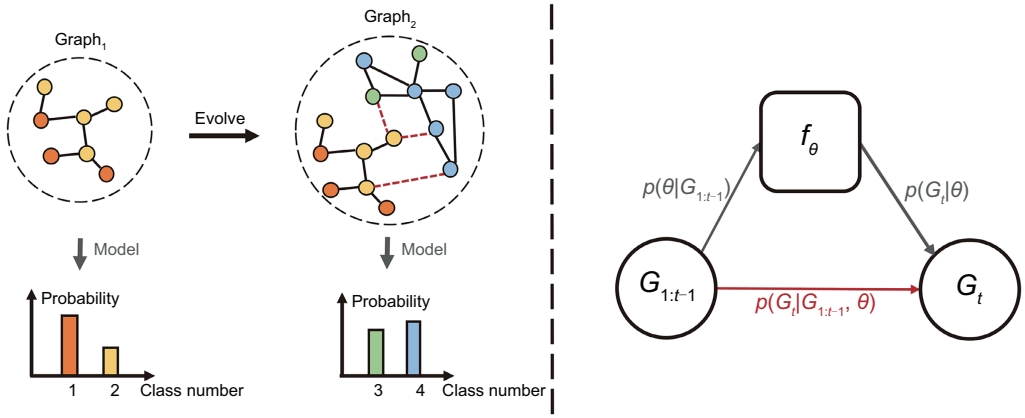

Fig. 1 Left: visualization of continual graph learning (CGL). Right: illustration of conditional probabilities for CGL. The gray lines show Bayes' rule for independent identically distributed data. The red line represents the influence of previous data on the current graph. References to color refer to the online version of this figure

To enhance training efficiency, we employ a simple yet effective multi-layer perceptron (MLP) model that shares the same weight space as its counterpart GNN during training, eliminating the time-consuming message-passing process. During inference, we use graph structure information to enhance prediction performance. Our method achieves state-of-the-art performance in both average accuracy (AA) and average forgetting (AF) across four large-scale CGL datasets in two challenging settings, effectively improving model performance while reducing catastrophic forgetting. Furthermore, compared to GNN-based methods, E-CGL achieves an average training/inference speedup of 15.83×/4.89×, with a maximum increase of 28.44×/8.89× on the OGBN-Products dataset.

The contributions of this paper can be summarized as follows:

1. Problem characterization. We identify and formally define two key challenges in CGL: topological interdependence, where evolving graph snapshots share structure and attributes, and computational efficiency, which arises from the need to process large, dynamically growing graphs efficiently.

2. Interdependence solution. We develop graph-aware replay strategies with importance and diversity sampling to explicitly capture inter-task dependencies. These strategies preserve critical information across updates.

3. Efficiency solution. We design an efficient graph learner with a shared-weight MLP encoder that accelerates training by up to 28.4× and reintroduces structural information during inference with the minimal performance compromises.

4. Empirical validation. We validate E-CGL on four datasets for supervised node classification using both task-incremental (task-IL) and class-incremental (class-IL) settings, achieving state-of-the-art performance in accuracy, forgetting, and computational efficiency.

2 Related works

CL aims to progressively acquire and integrate knowledge from new tasks while retaining previously learned information. Retraining the model with both old and new data is a naive approach to CL, but it is impractical in real-world applications since it is time-consuming, labor-intensive, and expensive. Another approach is to fine-tune the model using only new data; however, this often leads to catastrophic forgetting, where the model rapidly loses its ability to distinguish previously learned classes. Therefore, both training efficiency and robustness against catastrophic forgetting are crucial for successful CL.

Existing approaches to CL are categorized into three groups: regularization-, replay-, and architecture-based methods.

Regularization-based methods mitigate forgetting by introducing penalty terms that constrain parameter updates and help preserve past knowledge. For instance, learning without forgetting (LwF) (Li and Hoiem, 2018) uses knowledge distillation to regularize model parameters by reusing earlier model outputs as soft labels for current tasks. Elastic weight consolidation (EWC) (Kirkpatrick et al., 2017) directly adds quadratic penalties to model weights to prevent drastic parameter shifts. The approach named memory aware synapses (MAS) (Aljundi et al., 2018) follows a similar principle to EWC, but it measures the sensitivity of predictions to parameters and uses it as the regularization term.

Replay-based methods mitigate forgetting by maintaining a small subset of previous task data to retrain the model alongside new data. For example, gradient episodic memory (GEM) (Lopez-Paz and Ranzato, 2017) retains representative samples in episodic memory, and adjusts the gradients of the current task to prevent forgetting.

Architecture-based approaches allocate separate parameter subsets for different tasks to avoid drastic changes. While most of these methods can be adapted to graph learning with proper modifications, they often fail to account for the unique properties of graph data, resulting in performance degradation.

With increased interest in CGL, several methods tailored for graph-structured data have emerged. Topology-aware weight preserving (TWP) (Liu HH et al., 2021) is a regularization-based technique that uses penalization to maintain topological information from previous graphs. Despite their simplicity, regularization-based methods often face a trade-off between retaining old knowledge and learning new patterns due to limited model capacity.

Replay-based approaches have gained popularity in CGL. Continual graph neural network (ContinualGNN) (Wang JS et al., 2020) uses a replay-based

approach to sample nodes, balancing the findings of novel patterns with the preservation of old ones. Experience replay-based graph neural network (ER-GNN) (Zhou and Cao, 2021) includes three replay strategies tailored for GNNs. However, these node sampling techniques are designed for independent and identically distributed (IID) data, and do not explicitly consider graph topology. Sparsified subgraph memory (SSM) (Zhang et al., 2022b) reduces memory consumption by sparsifying replayed graphs. Similarly, condense and train (CaT) (Liu YL et al., 2023) uses graph condensation techniques to generate synthetic replayed graphs. Topology-aware graph coarsening and continual learning (TACO) (Han XX et al., 2024) develops a graph coarsening algorithm based on node representation proximities to reduce graph size during memory replay. Continual learning framework specifically designed for dynamic network embedding (CLDNE) (Wang ZZ et al., 2025) uses a streaming graph autoencoder to capture both global and local patterns, and experience replay and knowledge distillation to mitigate forgetting. Diversity enhancement and structure learning for rehearsal (DSLR) (Choi et al., 2024) considers both class representativeness and diversity for replayed nodes. Mapping-aware graph condensation (MCond)(Gao et al., 2024) creates a small synthetic graph to preserve training performance and learns one-to-many mappings from the original to synthetic nodes using transductive and inductive loss terms.

Architecture-based methods include dynamic graph incremental learning (DyGRAIN) (Kim et al., 2022), which uses time-varying receptive fields to capture temporal patterns. Graph continual learning (GCL) (Rakaraddi et al., 2022) combines architecture and replay techniques, using reinforcement learning to adapt the GNNs' capacity to task demands. In addition, continual graph learning benchmark (CGLB) (Zhang et al., 2022a) provides benchmark tasks and evaluation protocols for CGL, highlighting the importance of modeling both inter-task and intra-task edge dynamics.

Despite growing progress, the area of CGL still remains underexplored, with numerous open challenges. Beyond the interdependency and efficiency issues that we seek to tackle in this paper, other open challenges include: (1) understanding the impact of neighborhood evolution on CL, (2) handling the complexities of CL on heterogeneous graphs, (3) developing strategies for actively forgetting obsolete patterns, and more. Further research is needed to systematically address these challenges and advance our understanding of CL on graph-structured data.

3 Methodology

In this section, we delve into the technical details of E-CGL. First, we provide the fundamental concepts of CGL. Then, we discuss two essential components of our approach: graph-dependent replay and efficient graph learner. Finally, we conduct a complexity analysis to demonstrate the efficiency of E-CGL.

3.1 Preliminaries

Notations: A growing graph can be decomposed into a series of dynamic graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ based on timestamps. Each subgraph $G_t = (\mathcal{V}_t, \mathcal{E}_t)$ constitutes a distinct task t and has different types of nodes $\mathcal{V}_t$ and edges $\mathcal{E}_t$. In the context of node classification tasks in a CL setting, we denote $\boldsymbol{A}_t \in \mathbb{R}^{N_t \times N_t}$ as the adjacency matrix, $\boldsymbol{X}_t \in \mathbb{R}^{N_t \times K}$ as the raw node features, and $\boldsymbol{Y}_t \in \mathbb{R}^{N_t \times C}$ as the one-hot node labels for G_t. Here, $N_t = |\mathcal{V}_t|$ represents the number of nodes, K denotes the dimension of raw features, $\mathbb{R}$ denotes the set of real numbers, and C represents the total number of classes. Since the methodologies are all based on the current task t, unless otherwise specified, we will omit the subscript t for brevity.

Problem definition: In the CGL setting, each subgraph G_t in $\mathcal{G}$ has no overlap in category labels, and only the data at current time t are visible to the model due to the storage limitation. In what follows, task t and time t share the same meaning. The goal is to learn the newly emerging information while preventing catastrophic forgetting of previous knowledge. For each subgraph G_t, we split it into the training set $G_t^{\mathrm{tr}} = (\mathcal{V}_t^{\mathrm{tr}}, \mathcal{E}_t^{\mathrm{tr}})$ and the test set $G_t^{\mathrm{te}} = (\mathcal{V}_t^{\mathrm{te}}, \mathcal{E}_t^{\mathrm{te}})$ to train and evaluate the current model f, which is parameterized by θ_t.

3.2 Graph-dependent replay

As mentioned in Section 1, CGL faces novel challenges for alleviating catastrophic forgetting due

to the topological dependencies of nodes and graphs. To overcome these challenges, a direct solution is to replay historical data and rebuild such dependencies. To justify the replay-based strategy from a probabilistic perspective, we compute the posterior probability $p(\theta|\mathcal{D})$ by applying Bayes' rule to the prior probability of the parameters $p(\theta)$ and likelihood function $p(\mathcal{D}|\theta)$:

$$\ln p(\theta|\mathcal{D}) = \ln p(\mathcal{D}|\theta) + \ln p(\theta) - \ln p(\mathcal{D}), \tag{2}$$

where $\mathcal{D}$ represents the dataset. By rearranging Eq. (2) and taking into account the combination of new dataset $\mathcal{D}_{\text{new}}$ and previous dataset $\mathcal{D}_{\text{old}}$, we have

$$\begin{aligned}&\ln p(\theta|\mathcal{D}_{\text{new}} \cup \mathcal{D}_{\text{old}})\\ &= \ln p(\mathcal{D}_{\text{new}}|\theta, \mathcal{D}_{\text{old}}) + \ln p(\theta|\mathcal{D}_{\text{old}}) - \ln p(\mathcal{D}_{\text{new}}).\end{aligned} \tag{3}$$

The first term $\ln p(\mathcal{D}_{\text{new}}|\theta, \mathcal{D}_{\text{old}})$ on the right-hand side shows the interdependencies of graph data, indicating that the distribution of new data can be influenced by the old data. Such interdependencies make the problem challenging to address if the original dataset $\mathcal{D}_{\text{old}}$ is unavailable. Based on the above analysis, we choose replay-based methods to prevent catastrophic forgetting in the CGL scenario. In general, the replay-based strategy maintains a memory buffer $\mathcal{M}$ to store historical data from previous tasks. When a new task t arrives, the model learns from the new data while replaying from the old data:

$$\mathcal{L}_{\text{new}} = \sum_{i \in \mathcal{V}^{\text{tr}}} L_{\text{CE}}(f(\boldsymbol{x}_i), \boldsymbol{y}_i), \tag{4}$$

$$\mathcal{L}_{\text{replay}} = \sum_{j \in \mathcal{M}} L_{\text{CE}}(f(\boldsymbol{x}_j), \boldsymbol{y}_j), \tag{5}$$

where $L_{\text{CE}}(f(\boldsymbol{x}_i), \boldsymbol{y}_i) = -\boldsymbol{y}_i \ln f(\boldsymbol{x}_i)$ represents the cross-entropy loss, $\boldsymbol{x}_i$ is the feature vector of input node i, and $\boldsymbol{y}_i$ is the label vector of node i. Due to storage limit, it is impractical to replay all historical data; hence, effective sampling strategies are needed. In this paper, we present a novel sampling strategy that considers both the importance sampling and diversity sampling of graph data, using their topological and attributive characteristics.

1. Importance sampling

Google proposed PageRank (Page et al., 1999) as a web page ranking algorithm. It evaluates the importance of each node by considering the importance of other nodes linked to it. Formally, it iteratively updates the ranking scores using the first-order Markov chain until convergence:

$$\boldsymbol{\pi} = d\boldsymbol{T}\boldsymbol{\pi} + \frac{1-d}{N}\mathbf{1}, \tag{6}$$

$$T_{ij} = \begin{cases} \frac{1}{\delta_j}, & \text{if directed edges } (j,i) \in \mathcal{E}, \\ \frac{1}{N}, & \text{if } \delta_j = 0, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $\boldsymbol{\pi} \in \mathbb{R}^{N\times 1}$ denotes the ranking score for N nodes, d is the damping factor, $\boldsymbol{T}$ is the transition matrix based on graph topology, δ_j is the out-degree of node j, and $\mathbf{1}$ is a column vector with all elements equal to 1. Under the Markov chain framework, the ranking score $\boldsymbol{\pi}$ can be interpreted as the transition probability distribution of a random walk on the graph. Nevertheless, a major defect of PageRank is that it only considers the topological structures of a graph, while neglecting the node attribute information. This could be problematic when dealing with large-scale graphs with rich attributes. Inspired by prior studies (Hsu et al., 2017), we propose to use the node attributes as well as the graph topology for importance ranking.

Consider the homophilous assumption (Zhu et al., 2020), which states that nodes with similar attributes tend to share similar labels and importance. As a result, we use another random walk based on the attribute similarity matrix as transition probability. Defining a fully connected version of the current graph G^{fc}, whose edge weight is the similarity of its connected nodes $s(i,j)$, then the transition matrix of G^{fc} is given as

$$Q_{ij} = \frac{s(i,j)}{\sum_{k\in\mathcal{V}} s(k,j)}, \tag{8}$$

wherein we use radial basis function (RBF) (Rahimi and Recht, 2007) as the similarity metric: $s(i,j) = \mathrm{e}^{-\gamma||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}$. In this way, the value of Q_{ij} is guaranteed in $[0,1]$. With the attribute-based transition matrix and the topology-based transition matrix of vanilla PageRank, the updating strategy for importance ranking is derived as

$$\boldsymbol{\pi}_{\text{Imp}} = d_{\text{c}}\boldsymbol{T}\boldsymbol{\pi}_{\text{Imp}} + (1-d_{\text{c}})\boldsymbol{Q}\boldsymbol{\pi}_{\text{Imp}}, \tag{9}$$

where $d_{\text{c}} \in [0,1]$ is the controlling parameter that balances the contribution of $\boldsymbol{T}$ and $\boldsymbol{Q}$ to $\boldsymbol{\pi}_{\text{Imp}}$. However, the computation of $\boldsymbol{Q}$ and update of $\boldsymbol{Q}\boldsymbol{\pi}_{\text{Imp}}$ take the time complexity of $O(|\mathcal{V}|^2)$, which is unacceptable for large graphs. To solve the issue, we use

an approximation trick (Hsu et al., 2017) to simplify the process. First we define a vector as follows:

$$r_i = \frac{1}{z}\sum_{j\in\mathcal{V}} s(i,j), \tag{10}$$

where $z = \sum_i \sum_j s(i,j)$ is the normalization term. From the definition we obtain

$$\mathbf{1}\cdot\boldsymbol{r} = \boldsymbol{Q}\boldsymbol{r}. \tag{11}$$

The proof for Eq. (11) is provided in the supplementary materials. From Eq. (11), we find vector $\boldsymbol{r}$ serves as the stationary probability distribution for $\boldsymbol{Q}$, as well as the corresponding eigenvector of the largest eigenvalue 1 of $\boldsymbol{Q}$. Therefore, we use $\boldsymbol{r}$ as the surrogate of $\boldsymbol{Q}\boldsymbol{\pi}_{\mathrm{Imp}}$, and Eq. (9) is reformed as

$$\boldsymbol{\pi}_{\mathrm{Imp}} = d\boldsymbol{T}\boldsymbol{\pi}_{\mathrm{Imp}} + (1-d)\boldsymbol{r}. \tag{12}$$

With $\boldsymbol{r}$ to replace $\boldsymbol{Q}\boldsymbol{\pi}_{\mathrm{Imp}}$ and RBF as similarity, the importance ranking complexity in Eq. (12) can be further reduced to $O(|\mathcal{V}|K^2)$ by Taylor expansion $\mathrm{e}^x \approx 1 + x + \frac{1}{2}x^2 (x \to 0)$. We defer the detailed derivation in the supplementary materials.

2. Diversity sampling

The algorithm mentioned above selects important nodes to counteract catastrophic forgetting. However, it is equally important to consider instances that are vulnerable to subsequent tasks. The influence on the old graph could lead to the emergence of new patterns while diminishing existing ones. To capture these patterns, we propose a heuristic strategy that samples nodes to enhance the diversity of the memory buffer.

Specifically, we define nodes as diverse if they significantly differ from their surrounding nodes at the feature level. To determine diversity, we calculate the average feature of a node's neighbors (using only 1-hop neighbors for efficiency), and then compare it with the feature of the node itself:

$$\pi_{\mathrm{Div}}(i) = \left|\left| \boldsymbol{x}_i - \frac{1}{|\mathcal{N}_i|}\sum_{u\in\mathcal{N}_i} \boldsymbol{x}_u \right|\right|. \tag{13}$$

$\pi_{\mathrm{Div}}(i)$ reflects the diversity between node i and its surrounding environment, and $|\mathcal{N}_i|$ represents the number of node i's neighborhoods. Nodes with higher diversity may be vulnerable to new patterns and should be replayed in the future. Overall, the memory buffer $\mathcal{M}$ is updated:

$$\mathcal{M} = \mathcal{M} \cup \mathrm{argtopk}(\boldsymbol{\pi}_{\mathrm{Imp}}) \cup \mathrm{argtopk}(\boldsymbol{\pi}_{\mathrm{Div}}), \tag{14}$$

where $\mathrm{argtopk}(\boldsymbol{\pi}_{\mathrm{Imp}})$ and $\mathrm{argtopk}(\boldsymbol{\pi}_{\mathrm{Div}})$ refer to the indices of the top k elements of vectors $\boldsymbol{\pi}_{\mathrm{Imp}}$ and $\boldsymbol{\pi}_{\mathrm{Div}}$, respectively.

3.3 Efficient graph learner

To address the efficiency challenge in practical applications involving large growing graphs, we propose an efficient graph learner. This approach involves incorporating an MLP for training and using its weights to initialize GNNs for inference tasks. Specifically, a typical graph convolution network (GCN) (Kipf and Welling, 2017) layer can be formulated as

$$\boldsymbol{H}^{(l)} = \sigma\left(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}_{\mathrm{GCN}}\right), \tag{15}$$

where σ represents the nonlinear activation function, $\tilde{\boldsymbol{A}}$ represents the normalized adjacency matrix (Kipf and Welling, 2017), $\boldsymbol{H}^{(l-1)}$ denotes the $(l-1)^{\mathrm{th}}$ layer of the graph convolution, $\boldsymbol{H}^{(0)} = \boldsymbol{X}$ is the input node feature, and $\boldsymbol{W}^{(l)}_{\mathrm{GCN}}$ is the learnable weight matrix of the l^{th} layer. As illustrated in Fig. 2, we can further decouple the formulation into two operations: message passing and feature transformation (Wu et al., 2019):

$$\text{Message passing: } \hat{\boldsymbol{H}}^{(l-1)} = \tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}, \tag{16}$$

$$\text{Feature transformation: } \boldsymbol{H}^{(l)} = \sigma\left(\hat{\boldsymbol{H}}^{(l-1)}\boldsymbol{W}^{(l)}_{\mathrm{GCN}}\right). \tag{17}$$

The message passing mechanism, which aggregates information from each node's neighborhood, has long been considered as the core part for GNN models, e.g., GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018). However, sparse matrix multiplication accounts for the majority of computing time. Recent studies (Han XT et al., 2023; Yang et al., 2023) have found that the effect of message passing mainly comes from its generalization ability in inference, rather than its representation ability of learning better features in training. This finding opens up the possibility of removing message passing during training to improve efficiency. Furthermore, message passing is non-parametric in most cases, which means that it can be plug-and-play integrated into existing models without the need for training. As a result, Eq. (15) degrades to a simple layer of MLP

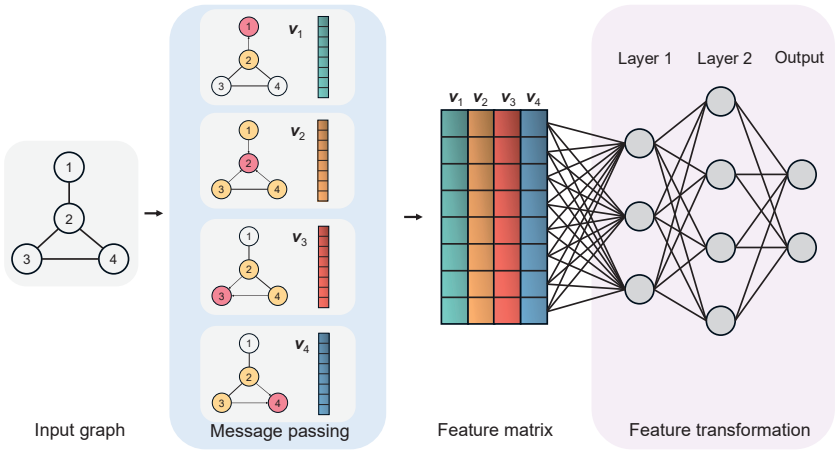

Fig. 2 Architecture of GCN, which consists of the message passing and feature transformation processes. Our E-CGL removes the message passing process during training, simplifying the training process to an MLP network

by removing message passing:

$$\boldsymbol{H}^{(l)} = \sigma\left(\boldsymbol{H}^{(l-1)}\boldsymbol{W}_{\mathrm{MLP}}^{(l)}\right). \tag{18}$$

Note that when the dimensions of hidden layers are set the same for $\boldsymbol{W}_{\mathrm{MLP}}$ and $\boldsymbol{W}_{\mathrm{GCN}}$, they will have identical weight space, allowing the weights of an MLP and its counterpart GCN to be transferred to each other. Based on this premise, we propose an efficient graph learner for fast model adaptation under a continual setting. Concretely, we first remove the message passing scheme of a GCN and initialize its counterpart MLP network. The parameters are then optimized using $\boldsymbol{W}_{\mathrm{MLP}}$ with only the node features as input, which speeds up the training process by avoiding sparse matrix multiplication. Formally, the training objective of Eq. (5) with MLP is derived as follows:

$$\mathcal{L}_{\mathrm{new}} = \sum_{i\in\mathcal{V}^{\mathrm{tr}}} L_{\mathrm{CE}}(f_{\mathrm{MLP}}(\boldsymbol{x}_i;\boldsymbol{W}_{\mathrm{MLP}}),\boldsymbol{y}_i), \tag{19}$$

$$\mathcal{L}_{\mathrm{replay}} = \sum_{j\in\mathcal{M}} L_{\mathrm{CE}}(f_{\mathrm{MLP}}(\boldsymbol{x}_j;\boldsymbol{W}_{\mathrm{MLP}}),\boldsymbol{y}_j). \tag{20}$$

During inference, we use the trained $\boldsymbol{W}_{\mathrm{MLP}}$ as the parameters of the corresponding GCN model, and the topological information is used again with message passing added:

$$\boldsymbol{H}^{(l)} = \sigma\left(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}_{\mathrm{MLP}}^{(l)}\right), \tag{21}$$

$$\hat{\boldsymbol{Y}} = \mathrm{softmax}(\boldsymbol{H}^{(L)}). \tag{22}$$

Here, L stands for the last layer of features; i.e., the output of the model. $\hat{\boldsymbol{Y}}$ is the labeling matrix obtained from the model prediction. Note that the GCN encoder can be generalized to any message-passing-based GNN with proper modifications; we use GCN here for notation simplicity.

3.4 Training objective

The pseudo-code of E-CGL, which combines the graph-dependent replay module and the efficient graph learner module, is described in the supplementary materials. The overall training objective for E-CGL at each task is formulated as

$$\mathcal{L} = \mathcal{L}_{\mathrm{new}} + \lambda\mathcal{L}_{\mathrm{replay}}, \tag{23}$$

where λ represents the weight that balances the strength of two losses, and we simply set it as 1.

3.5 Complexity analysis

1. Time complexity

The training process of efficient graph learner is equivalent to that of an L-layer MLP network. For simplicity, we assume that the dimension of the node attributes is equal to the hidden size of the MLP, denoted by K. The time complexity of the efficient graph learner is $O(L|\mathcal{V}|K^2)$, which is significantly smaller than $O(L|\mathcal{V}|^2K + L|\mathcal{V}|K^2)$ of GNNs.

The sampling step has a lower overhead than model training because it is only executed once per task. The time complexity of importance sampling has been optimized to $O(|\mathcal{V}|K^2)$, according to the derivation in the supplementary materials. Diversity sampling takes $O(|\mathcal{V}|\delta K)$, where δ represents

the mean degree of each node. Both complexities are linear with respect to $|\mathcal{V}|$; therefore, they are suitable for large graphs.

2. Space complexity

Because the efficient graph learner does not use graph structure, its space complexity is $O(L|\mathcal{V}|K + LK^2)$, making it more memory-efficient than other GNN-based methods with $O(|\mathcal{V}|^2 + L|\mathcal{V}|K + LK^2)$. As a result, our method can allow large batch sizes or even the whole batch training for large graphs, and we will empirically study the time and space complexity of E-CGL in the next section.

4 Experiments and results

In this section, we present comprehensive experimental results, including settings, quantitative evaluations under both task-IL and class-IL scenarios, analysis of runtime and GPU memory usage, and ablation studies. Additional experimental details and results analysis are provided in the supplementary materials.

4.1 Experimental settings

1. Task-IL and class-IL settings

CGLB's benchmark study (Zhang et al., 2022a) proposed two settings for CGL: task-IL and class-IL. In the task-IL setting, a task indicator helps the model in distinguishing between classes within each task. In contrast, in the class-IL setting, task indicators are not provided, requiring the model to differentiate among all classes from both the current and previous tasks.

To illustrate these settings, we follow the examples from their studies. Consider a model trained on a citation network and given two tasks: (physics, chemistry) and (biology, math). In the task-IL setting, each document comes with a task indicator indicating whether it belongs to (physics, chemistry) or (biology, math). Consequently, the model only needs to categorize the document into one of the two task categories without distinguishing among all four classes. Conversely, in the class-IL setting, a document may belong to any of the four classes, and the model must categorize it appropriately. The class-IL setting is generally more challenging, as the model must handle an increasing number of classes over time. Without the loss of generality, we conduct experiments under both task-IL and class-IL settings.

2. Datasets

We conduct experiments on four node classification datasets: CoraFull (Bojchevski and Günnemann, 2018), Reddit (Hamilton et al., 2017), OGBN-Arxiv (Hu WH et al., 2020), and OGBN-Products (Hu WH et al., 2020). Following the methodology outlined by CGLB (Zhang et al., 2022a), we partition the label space of the datasets into several non-overlapping segments to simulate the task/class-IL setting. For example, in the case of CoraFull, we divide the original dataset into 14 tasks, each comprising a five-way node classification task. The detailed dataset statistics are presented in the supplementary materials.

3. Baselines

We compare E-CGL against 12 state-of-the-art CL techniques. These include four conventional methods (LwF (Li and Hoiem, 2018), EWC (Kirkpatrick et al., 2017), MAS (Aljundi et al., 2018), and GEM (Lopez-Paz and Ranzato, 2017)) originally designed for computer vision (CV) tasks, along with eight methods specifically tailored for graph tasks (TWP (Liu HH et al., 2021), ER-GNN (Zhou and Cao, 2021), DyGRAIN (Kim et al., 2022), SSM (Zhang et al., 2022b), CaT (Liu YL et al., 2023), GCL (Rakaraddi et al., 2022), TACO (Han XX et al., 2024), and DSLR (Choi et al., 2024)). We also incorporate fine-tuning and joint training as performance lower-bound and upper-bound baselines, respectively.

4. Metrics

Following the definitions in CGLB (Zhang et al., 2022a), we use the performance matrix $\boldsymbol{M}^{\mathrm{P}} \in \mathbb{R}^{T\times T}$ to depict the dynamics of the overall performance, where M^{P}_{ij} represents the accuracy of task j after the model has been trained on task i. Based on this performance matrix, we compute the AA and AF (Lopez-Paz and Ranzato, 2017):

$$\mathrm{AA} = \frac{\sum_{j=1}^{i} M^{\mathrm{P}}_{ij}}{i}, \quad i = 1, 2, \ldots, T,$$

$$\mathrm{AF} = \frac{\sum_{j=1}^{i-1} M^{\mathrm{P}}_{ij} - M^{\mathrm{P}}_{jj}}{i-1}, \quad i = 2, 3, \ldots, T.$$

It is important to note that in CL scenarios, AF is typically negative, indicating the occurrence of catastrophic forgetting.

4.2 Task-IL results and analysis

The results of task-IL node classification are shown in Table 1. Our proposed E-CGL method provides state-of-the-art performance, with the highest

Table 1 Results on node classification tasks under task-IL setting

Category	Method	CoraFull		OGBN-Arxiv		Reddit		OGBN-Products	
		AA (%)	AF (%)	AA (%)	AF (%)	AA (%)	AF (%)	AA (%)	AF (%)
Lower-bound	Fine-tuning	39.5±1.8	−54.6±1.7	51.5±4.6	−34.7±5.0	57.9±3.5	−44.9±3.9	68.7±2.0	−27.1±2.4
Upper-bound	Joint training	91.0±0.2	N/A	83.8±0.5	N/A	98.2±0.1	N/A	94.4±0.3	N/A
CV-based	LwF	56.7±5.9	−35.2±6.4	79.2±0.7	−2.1±0.8	72.6±6.4	−28.1±7.2	73.8±2.0	−22.4±1.8
	EWC	66.4±3.3	−24.3±3.6	62.3±4.3	−16.3±4.5	87.5±4.4	−11.3±5.4	92.3±1.4	−1.0±0.3
	MAS	86.4±0.8	<u>−0.2</u>±0.3	<u>81.0</u>±0.8	<u>0.0</u>±0.0	89.7±1.7	**0.0**±0.2	83.6±1.0	**−0.1**±0.1
	GEM	81.1±0.8	−8.8±1.1	67.9±0.4	−9.3±0.6	55.6±23.3	−4.9±3.9	78.8±15.3	−1.7±3.0
Graph-based	TWP	86.4±0.4	−1.8±0.7	80.5±0.8	−1.7±0.8	87.8±5.5	−11.5±6.1	93.1±1.5	−1.2±0.9
	ER-GNN	88.9±0.1	**0.1**±0.3	75.0±0.3	−8.2±0.5	89.8±0.6	−9.2±0.8	<u>93.3</u>±0.7	<u>−0.9</u>±0.1
	DyGRAIN	N/A	N/A	70.9±0.3	−4.6±0.1	92.1±0.5	−3.5±0.1	73.4±0.2	−3.3±0.1
	SSM	88.9±0.5	−0.8±0.1	80.8±1.3	−2.4±0.7	**93.7**±0.3	−4.9±0.6	91.6±0.6	−3.6±0.2
	CaT	84.3±0.5	−6.1±0.8	70.6±4.2	−14.2±2.3	90.9±0.3	−5.5±0.5	OOM	OOM
	GCL-SAGE	87.1±0.7	−4.5±0.2	77.9±1.2	−12.4±4.0	91.0±0.4	−5.8±0.6	89.5±0.5	−3.9±0.4
	TACO	85.2±0.6	−6.6±0.9	79.3±1.5	−15.3±1.2	89.5±0.7	−7.0±1.8	87.2±0.5	-4.9±0.6
	DSLR	<u>89.1</u>±0.2	−1.5±0.3	80.4±0.9	−3.2±0.5	91.9±0.4	<u>−2.3</u>±0.4	92.7±0.3	−1.5±0.2
Ours	E-CGL	**89.6**±0.1	−2.5±0.2	**82.1**±1.0	**0.2**±0.2	<u>92.2</u>±0.7	−2.7±0.8	**93.9**±0.6	−1.2±0.3

For joint training, it is trained on all tasks as the upper bound; therefore, we do not provide its AF values. DyGRAIN has no open-source code, and thus we use the results reported in the reference (Kim et al., 2022). Negative values for AF indicate performance drop due to catastrophic forgetting, and N/A refers to not available. The best results are marked in bold, and the second-best are underlined

AA on CoraFull (89.6%), OGBN-Arxiv (82.1%), and OGBN-Products (93.9%), while ranking second on Reddit (92.2%) behind SSM (93.7%). Remarkably, E-CGL achieves a positive AF value (+0.2%) on OGBN-Arxiv, indicating forward knowledge transfer, while maintaining moderate forgetting (−2.7% to −1.2%) on other datasets. This is particularly impressive given that E-CGL uses a simple MLP architecture to outperform sophisticated GNN competitors.

We compared the performance of different methods with the lower bound (fine-tuning) and upper bound (joint training), separately. Almost all methods fall within these bounds, showing varied degrees of improvement over fine-tuning. However, most methods still suffer from catastrophic forgetting, as reflected by negative AF values.

The comparison between method categories reveals two key insights: (1) Graph-based methods (DSLR, ER-GNN, and SSM) dominate the top rankings, with DSLR showing particularly strong performance (89.1% AA on CoraFull and −2.3% AF on Reddit); (2) While CV-based methods achieve low forgetting (such as MAS with −0.2% to 0.0% AF across datasets), they suffer from significant accuracy drops when compared to graph-based approaches. The performance gap between graph-based methods and CV-based baselines (e.g., 89.6% vs. 86.4% on CoraFull) emphasizes the representation gap between graph data and Euclidean data, emphasizing the importance of tailored approaches for graph-related tasks.

4.3 Class-IL results and analysis

Table 2 shows the experimental results under the class-IL setting, which indicate four key observations. First, all methods exhibit significantly degraded performance compared to the task-IL setting. This catastrophic performance collapse stems from the expanded label space in class-IL (e.g., 70-class prediction on CoraFull vs. 5-class in task-IL), which overwhelms conventional CV-based methods. The performance of these methods collapses to near fine-tuning levels (e.g., −91.2% to −86.9% AF on CoraFull), revealing their inadequacy for graph-structured class-incremental learning.

Second, the regularization-based method TWP performs poorly and is nearly indistinguishable from the fine-tuning baseline. This may be attributed to strong distributional differences between tasks, where regularization approaches struggle to retain performance on previous tasks without access to historical data. This underscores the effectiveness of replay-based methods for CGL.

Third, the performance of replay-based methods varies significantly across datasets: (1) ER-GNN's

Table 2 Results on node classification tasks under class-IL setting

Category	Method	CoraFull		OGBN-Arxiv		Reddit		OGBN-Products	
		AA (%)	AF (%)	AA (%)	AF (%)	AA (%)	AF (%)	AA (%)	AF (%)
Lower-bound	Fine-tuning	5.5±0.0	−90.7±0.3	8.5±0.3	−80.9±0.5	11.6±1.3	−93.6±2.5	4.4±1.4	−83.7±3.5
Upper-bound	Joint training	79.5±0.1	N/A	51.2±0.2	N/A	97.2±0.5	N/A	76.3±0.2	N/A
CV-based	LwF	5.9±0.5	−91.2±0.8	8.8±0.2	−81.3±0.5	11.8±0.4	−89.8±2.6	5.4±0.1	−88.7±1.1
	EWC	5.4±0.1	−91.1±3.4	8.8±0.2	−83.1±1.4	11.9±2.1	−95.3±1.7	3.5±0.3	−91.8±0.9
	MAS	4.3±0.1	−86.9±1.1	8.7±0.0	−77.0±0.7	9.1±0.2	−51.0±2.5	19.1±0.7	−16.2±0.1
	GEM	6.4±0.6	−89.9±0.9	8.8±0.1	−80.5±0.5	9.2±1.3	−11.2±4.8	7.7±3.0	**−15.3**±2.6
Graph-based	TWP	7.5±0.1	−86.3±0.5	8.5±0.1	−81.4±0.5	13.4±1.3	−93.6±1.4	3.3±0.6	−94.8±0.3
	ER-GNN	5.1±0.2	−90.7±0.2	17.3±0.8	−69.6±1.1	62.0±4.5	−39.4±5.1	6.8±1.6	−90.2±2.2
	SSM	**76.2**±1.2	**6.0**±0.1	33.0±0.4	**21.5**±0.4	74.8±1.7	**−7.1**±0.7	OOM	OOM
	CaT	14.8±5.2	−81.3±3.7	32.1±0.3	−48.8±0.5	74.1±2.2	−20.8±1.2	OOM	OOM
	GCL-SAGE	34.2±3.0	−17.7±5.3	23.5±1.8	−42.3±3.2	65.3±2.1	−25.1±2.5	34.8±1.5	−41.9±1.8
	TACO	71.5±1.3	−12.5±1.1	31.2±0.8	−35.6±0.9	73.2±1.5	−18.9±1.7	51.4±0.7	−28.3±0.6
	DSLR	75.8±0.7	−8.2±0.4	**34.5**±0.6	−28.1±0.5	77.9±0.8	−9.5±0.3	54.7±0.9	−19.8±0.4
Ours	E-CGL	68.2±0.2	−17.8±0.2	33.7±0.2	−41.0±0.1	**78.6**±1.0	−17.3±1.2	**56.3**±1.0	−31.0±1.1

For joint training, it is trained on all tasks as the upper bound; therefore, we do not provide its AF values. The results of DyGRAIN in the class-IL setting are not provided in the reference. Negative values for AF indicate performance drop due to catastrophic forgetting, and N/A refers to not available. The best results are marked in bold, and the second-best are underlined

performance is highly dependent on datasets. It achieves acceptable results on Reddit (62.0% AA) but poorly on CoraFull (5.1% AA) and OGBN-Products (6.8% AA). (2) SSM and CaT perform well on other datasets, but they fail to operate effectively on the large-scale OGBN-Products dataset, resulting in out-of-memory (OOM) errors. (3) In contrast, TACO, DSLR, and the proposed E-CGL demonstrate more stable and robust performance across all datasets.

Finally, joint training provides the best performance and remains the gold standard. This highlights that catastrophic forgetting remains a significant challenge in the class-IL setting. Although retraining on all historical data can mitigate forgetting, it has an unacceptable computational cost.

4.4 Runtime and memory usage analysis

To evaluate the efficiency of E-CGL, we conducted runtime and memory usage analysis, comparing our method with other graph CL methods on the task-IL setting.

Table 3 shows the average time for each training and inference epoch. We found that E-CGL significantly reduces both training and inference time when compared to its GCN version. E-CGL achieves an average speedup of 15.83 times during training and 4.89 times during inference. The magnitude of improvement improves as the dataset size increases. For example, the OGBN-Products dataset shows an improvement of 28.44 times during training and 8.89 times during inference. Additionally, E-CGL outperforms other CGL methods in both training and inference time.

Fig. 3 shows the maximum GPU memory usage for each method during training. We did not report the results of the OGBN-Products dataset since CaT (Liu YL et al., 2023) encountered an OOM error. It can be observed that the regularization-based TWP (Liu HH et al., 2021) has the lowest memory consumption because it does not need to explicitly store historical graph data. For the remaining replay-based methods, with the same

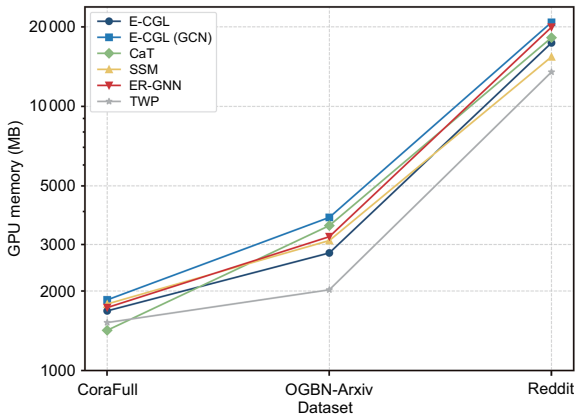

Fig. 3 GPU memory usage of different CGL methods under the task-IL setting

Table 3 Runtime (training/inference) comparison of different CGL methods under task-IL setting

Method	Runtime (ms)				Average time (ms)
	CoraFull	OGBN-Arxiv	Reddit	OGBN-Products	
TWP	34.25/105.71	37.33/23.49	354.68/1104.26	3217.46/2894.05	910.93/1031.88
ER-GNN	56.51/107.67	47.17/22.64	421.83/1075.20	4419.80/4385.23	1236.33/1397.69
SSM	52.68/109.89	49.63/27.65	491.99/1220.88	3976.74/4351.92	1142.76/1427.59
CaT	176.24/115.81	152.13/23.15	1719.85/1120.00	OOM	N/A
E-CGL (GCN)	28.94/110.79	43.18/27.39	1078.31/1032.93	5205.96/4912.76	1589.10/1520.97
E-CGL	**15.69**/**99.24**	**25.61**/**17.95**	**177.13**/**574.13**	**183.04**/**552.72**	**100.37**/**311.01**
Improvement	1.84×/1.12×	1.69×/1.53×	6.09×/1.80×	28.44×/8.89×	15.83×/4.89×

E-CGL (GCN) is the version that we use GCN as the encoder for both training and inference. Improvement indicates the performance improvement of our E-CGL compared with E-CGL (GCN). We report both training and inference time. The best results are marked in bold

sampling budget (1000 nodes), our proposed E-CGL exhibits relatively low memory usage, benefiting from its simple architecture. Additionally, compared to its GCN counterpart, the MLP version shows a significant reduction in memory consumption.

These efficiency improvements can be attributed to the design of the efficient graph learner, which eliminates the need for time-consuming message passing during training and allows direct training on large graphs without batching. The reduced training time of E-CGL makes it a practical and efficient solution for CGL tasks, allowing for faster experimentation, model iteration, and deployment.

4.5 Ablation studies

To assess the effectiveness of different components in E-CGL, we conducted ablation studies under the task-IL setting. Two key components were evaluated: the replay-based sampling strategy and the choice of training encoder (MLP vs. GCN). The results are shown in Table 4.

Replay-based sampling strategy. First, we examined the impact of removing the importance and diversity samplers, separately, while maintaining the same sampling budget for the other. The results indicate that when either sampler is removed, the performance of E-CGL decreases in terms of AA, emphasizing the necessity of both components in the proposed replay-based sampling strategy. Second, in most cases, removing the importance sampler results in a larger performance drop compared to removing the diversity sampler. Additionally, the magnitude of the difference reflects the dataset difficulty, which will be discussed in the visualization part of the supplementary materials. On simpler datasets such as OGBN-Products, there is minimal difference between the full version and versions with removed samplers; however, on relatively challenging datasets such as OGBN-Arxiv and Reddit, the sampling strategy has a more significant impact on the results.

MLP vs. GCN training encoders. Furthermore, we replaced the efficient graph learner (an MLP-based encoder) with a GCN as the training encoder, as shown in the last row of Table 4. As expected, using GCN leads to marginally higher performance in most cases, confirming that message passing enhances accuracy by leveraging graph

Table 4 Ablation study results of E-CGL

Ablated component	CoraFull		OGBN-Arxiv		Reddit		OGBN-Products	
	AA (%)	AF (%)	AA (%)	AF (%)	AA (%)	AF (%)	AA (%)	AF (%)
E-CGL	**89.6**±0.1	−2.5±0.2	82.1±1.0	0.2±0.1	92.2±0.7	−2.7±0.8	93.9±0.6	−1.2±0.3
Importance sampler	88.3±0.2	**1.5**±0.2	80.0±0.6	−3.2±0.5	88.9±2.0	**−1.4**±2.1	93.7±0.4	−0.3±0.1
Diversity sampler	89.3±0.4	−2.0±0.1	81.5±0.7	−1.0±0.1	90.6±1.8	−3.1±1.5	93.2±1.2	−2.5±0.9
−MLP, +GCN	88.8±0.7	−2.9±0.9	**82.7**±0.3	**0.8**±0.6	**95.8**±1.9	−1.6±1.2	**94.0**±0.6	**0.0**±0.4

We removed the proposed importance sampler and diversity sampler for replay, separately. "−MLP, +GCN" means that we employed GCN instead of MLP for training. Negative values for AF indicate performance drop due to catastrophic forgetting. The best results are marked in bold

structure. However, GCN has a significant computational cost, increasing training time by 15.83× on average compared to MLP (Table 3). Notably, because structural information is reintroduced during inference via the shared-weight GCN, E-CGL retains the benefits of graph-aware reasoning at test time, which partially mitigates the limitations of MLP training.

Specifically, on structurally complex graphs (e.g., OGBN-Arxiv or Reddit), GCN's accuracy gains are most pronounced (up to +3.6 percentage points AA), owing to their rich neighborhood patterns that MLPs cannot explicitly capture. Conversely, on simpler graphs such as OGBN-Products, where node features are highly predictive, the MLP-GCN gap narrows to only 0.1 percentage points AA. This shows that MLPs are sufficient when features are strongly correlated with labels.

Therefore, the choice between MLP and GCN depends on the deployment requirements. For latency-sensitive applications (e.g., real-time recommendations), MLP's faster training justifies its minor accuracy loss. Conversely, in accuracy-critical scenarios involving complex graph structures, GCN's gains may justify its computational overhead.

5 Conclusions

In this paper, we have addressed two key challenges in CGL: the interdependencies in graph data and the efficiency concerns associated with growing graphs. To tackle these issues, we introduce an E-CGL that uses a graph-specific sampling strategy for replay and an MLP encoder for efficient training. Our extensive empirical results demonstrate the effectiveness of our method in terms of both performance and efficiency.

Despite the increasing interest in CGL, several limitations remain to be addressed. For instance, there is a need to explore methods for safeguarding historical data privacy during the replay phase and actively forgetting outdated knowledge to enhance model adaptability. Moreover, further investigation is required to effectively handle heterogeneous graphs and improve performance on graph classification tasks. We hope that E-CGL represents a step forward in CGL and inspires future research to explore broader applications and develop more advanced techniques in this domain.

Contributors

Jianhao GUO designed the research and drafted the paper. Zixuan NI and Yun ZHU helped organize the paper. Jianhao GUO and Siliang TANG revised and finalized the paper.

Conflict of interest

The authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Aljundi R, Babiloni F, Elhoseiny M, et al., 2018. Memory aware synapses: learning what (not) to forget. Proc 15th European Conf on Computer Vision, p.144-161. https://doi.org/10.1007/978-3-030-01219-9_9

Bojchevski A, Günnemann S, 2018. Deep Gaussian embedding of graphs: unsupervised inductive learning via ranking. 6th Int Conf on Learning Representations.

Buzzega P, Boschini M, Porrello A, et al., 2020. Dark experience for general continual learning: a strong, simple baseline. Proc 34th Int Conf on Neural Information Processing Systems, Article 1335.

Choi S, Kim W, Kim S, et al., 2024. DSLR: diversity enhancement and structure learning for rehearsal-based graph continual learning. Proc ACM Web Conf, p.733-744. https://doi.org/10.1145/3589334.3645561

Fan ZH, Wei ZY, Chen JJ, et al., 2022. A unified continuous learning framework for multi-modal knowledge discovery and pre-training. https://arxiv.org/abs/2206.05555

Gao XY, Chen T, Zang YL, et al., 2024. Graph condensation for inductive node representation learning. IEEE 40th Int Conf on Data Engineering, p.3056-3069. https://doi.org/10.1109/ICDE60146.2024.00237

Hamilton WL, Ying R, Leskovec J, 2017. Inductive representation learning on large graphs. Proc 31st Int Conf on Neural Information Processing Systems, p.1025-1035.

Han XT, Zhao T, Liu YZ, et al., 2023. MLPInit: embarrassingly simple GNN training acceleration with MLP initialization. The 11th Int Conf on Learning Representations.

Han XX, Feng Z, Ning Y, 2024. A topology-aware graph coarsening framework for continual graph learning. Proc 38th Int Conf on Neural Information Processing Systems, Article 4212.

Hsu CC, Lai YA, Chen WH, et al., 2017. Unsupervised ranking using graph structures and node attributes. Proc 10th ACM Int Conf on Web Search and Data Mining, p.771-779. https://doi.org/10.1145/3018661.3018668

Hu WH, Fey M, Zitnik M, et al., 2020. Open graph benchmark: datasets for machine learning on graphs. Proc 34th Int Conf on Neural Information Processing Systems, Article 1855.

Hu XT, Tang KH, Miao CY, et al., 2021. Distilling causal effect of data in class-incremental learning. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.3956-3965. https://doi.org/10.1109/CVPR46437.2021.00395

Kim S, Yun S, Kang J, 2022. DyGRAIN: an incremental learning framework for dynamic graphs. Proc 31st Int Joint Conf on Artificial Intelligence, p.3157-3163. https://doi.org/10.24963/IJCAI.2022/438

Kipf TN, Welling M, 2017. Semi-supervised classification with graph convolutional networks. 5th Int Conf on Learning Representations, p.3157-3163.

Kirkpatrick J, Pascanu R, Rabinowitz N, et al., 2017. Overcoming catastrophic forgetting in neural networks. *Proc Natl Acad Sci USA*, 114(13):3521-3526. https://doi.org/10.1073/pnas.1611835114

Li ZZ, Hoiem D, 2018. Learning without forgetting. *IEEE Trans Patt Anal Mach Intell*, 40(12):2935-2947. https://doi.org/10.1109/TPAMI.2017.2773081

Liu HH, Yang YD, Wang XC, 2021. Overcoming catastrophic forgetting in graph neural networks. Proc AAAI Conf on Artificial Intelligence, p.8653-8661. https://doi.org/10.1609/aaai.v35i10.17049

Liu YL, Qiu RH, Huang Z, 2023. CaT: balanced continual graph learning with graph condensation. IEEE Int Conf on Data Mining, p.1157-1162. https://doi.org/10.1109/ICDM58522.2023.00141

Lopez-Paz D, Ranzato M, 2017. Gradient episodic memory for continual learning. Proc 31st Int Conf on Neural Information Processing Systems, p.6470-6479.

Ni ZX, Wei LH, Tang SL, et al., 2023. Continual vision-language representation learning with off-diagonal information. Proc 40th Int Conf on Machine Learning, Article 1087.

Niepert M, Ahmed M, Kutzkov K, 2016. Learning convolutional neural networks for graphs. Proc 33rd Int Conf on Machine Learning, p.2014-2023.

Page L, Brin S, Motwani R, et al., 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report, SIDL-WP-1999-0120. http://ilpubs.stanford.edu:8090/422/

Rahimi A, Recht B, 2007. Random features for large-scale kernel machines. Proc 21st Int Conf on Neural Information Processing Systems, p.1177-1184.

Rakaraddi A, Siew Kei L, Pratama M, et al., 2022. Reinforced continual learning for graphs. Proc 31st ACM Int Conf on Information & Knowledge Management, p.1666-1674. https://doi.org/10.1145/3511808.3557427

Rebuffi SA, Kolesnikov A, Sperl G, et al., 2017. iCaRL: incremental classifier and representation learning. IEEE Conf on Computer Vision and Pattern Recognition, p.5533-5542. https://doi.org/10.1109/CVPR.2017.587

Srinivasan T, Chang TY, Alva LP, et al., 2022. CLiMB: a continual learning benchmark for vision-and-language tasks. Proc 36th Int Conf on Neural Information Processing Systems, Article 2135.

Su JW, Zou DF, Wu C, 2025. On the limitation and experience replay for GNNs in continual learning. Proc 3rd Conf on Lifelong Learning Agents, p.342-366.

Thrun S, 1994. A lifelong learning perspective for mobile robot control. Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.22-30. https://doi.org/10.1109/IROS.1994.407413

Veličković P, Cucurull G, Casanova A, et al., 2018. Graph attention networks. https://doi.org/10.48550/arXiv.1710.10903

Wang JS, Song GJ, Wu Y, et al., 2020. Streaming graph neural networks via continual learning. Proc 29th ACM Int Conf on Information & Knowledge Management, p.1515-1524. https://doi.org/10.1145/3340531.3411963

Wang ZZ, Sun YY, Zhang XK, et al., 2025. Continual learning with high-order experience replay for dynamic network embedding. *Patt Recogn*, 159:111093. https://doi.org/10.1016/j.patcog.2024.111093

Wu F, Zhang TY, de Souza AHJr, et al., 2019. Simplifying graph convolutional networks. Proc 36th Int Conf on Machine Learning, p.6861-6871.

Xu K, Hu WH, Leskovec J, et al., 2019. How powerful are graph neural networks? 7th Int Conf on Learning Representations.

Yang CX, Wu QT, Wang JH, et al., 2023. Graph neural networks are inherently good generalizers: insights by bridging GNNs and MLPs. https://doi.org/10.48550/arXiv.2212.09034

Yuan Q, Guan SU, Ni P, et al., 2023. Continual graph learning: a survey. https://arxiv.org/abs/2301.12230

Zhang XK, Song DJ, Tao DC, 2022a. CGLB: benchmark tasks for continual graph learning. Proc 36th Int Conf on Neural Information Processing Systems, Article 945.

Zhang XK, Song DJ, Tao DC, 2022b. Sparsified subgraph memory for continual graph representation learning. IEEE Int Conf on Data Mining, p.1335-1340. https://doi.org/10.1109/ICDM54844.2022.00177

Zhou F, Cao CT, 2021. Overcoming catastrophic forgetting in graph neural networks with experience replay. Proc 35th AAAI Conf on Artificial Intelligence, p.4714-4722.

Zhu J, Yan YJ, Zhao LX, et al., 2020. Beyond homophily in graph neural networks: current limitations and effective designs. Proc 34th Int Conf on Neural Information Processing Systems, Article 653.

List of supplementary materials